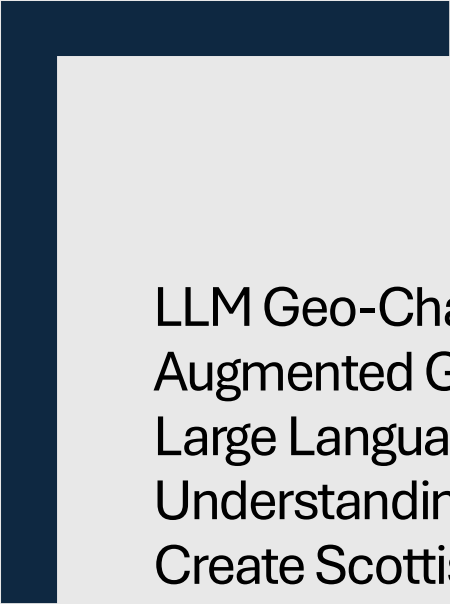


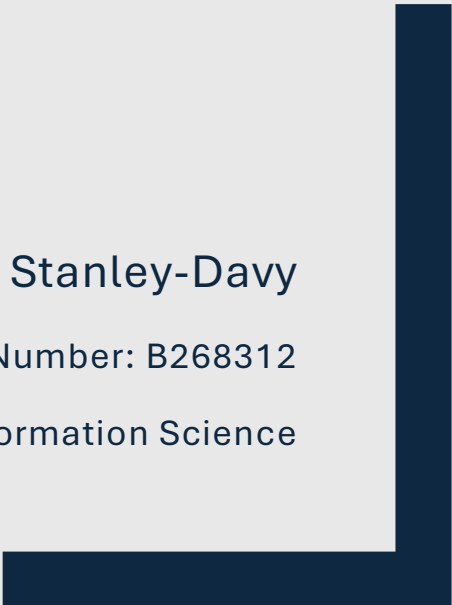


LLM Geo-Chat Interfaces: Designing A Retrieval Augmented Generation (RAG) Model to Improve Large Language Model (LLM) Geographical Understanding, Answer Spatial Queries and Create Scottish Maps

Emily Stanley-Davy

Exam Number: B268312

MSc Geographical Information Science



Statement of Originality

I declare that this dissertation represents my own work, and that where the work of others has been used it has been duly accredited. I further declare that the length of the components of this dissertation is 4,950 words for the Research Paper and 9,494 words for the Technical Report. Copyright of this dissertation is retained by the author and The University of Edinburgh. Ideas contained in this dissertation remain the intellectual property of the author and their supervisors, except where explicitly otherwise referenced. All rights reserved. The use of any part of this dissertation reproduced, transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise or stored in a retrieval system without the prior written consent of the author and The University of Edinburgh (Institute of Geography) is not permitted.

I agree that this dissertation and associated electronic documents, web pages, data, files and computer programs can be retained by the University. YES

I agree that, with the permission of my supervisor(s) or the Programme Director, these materials be made available for the purposes of preparing a publication. YES

I agree that, with the permission of my supervisor(s) or the Programme Director, these materials can be used within the University of Edinburgh for continued research or teaching. YES

Emily Stanley-Davy, 1st of August 2025

Acknowledgements

I would like to thank my supervisors Mark McLauchlan and Bruce Gittings for their support with this research. Thank you also to the Geosciences IT Department for their troubleshooting advice on the occasions when GeoBot went rogue.

Part I: Research Paper

Table of Contents

Abstract.....	3
Introduction.....	4
Research Questions.....	5
Aims.....	5
Objectives	5
Artificial Intelligence: Present Capabilities and Future Possibilities.....	6
Machine Learning and Large Language Models	6
Methodology.....	8
RAG Model Justification.....	8
Geographical Concepts.....	10
RAG Model Input Questions.....	12
Choice of RAG Model Platform.....	13
GeoBot: OpenAI RAG Model Summarised Methodology	14
No-Code Alternative RAG Chatbot	17
RAG Model Result Methodology.....	17
Results	19
Response Accuracy and Similarity	22
Response Clarity.....	26
Response Consistency.....	27
User Feedback Survey Results for User Satisfaction and Task Failure Rate	28
Discussion.....	30
Effectiveness of RAG Model Choice and Data on Enhancing LLM Scottish Geographical Responses	30
The Viability of Integrating Map Generation Capabilities into a RAG Model	32
GeoBot Geospatial Industry Application.....	32
Conclusion	34
Bibliography.....	35

Abstract

This research explores how RAG Models can improve the Scottish geographical queries of Large Language Models (LLMs). Specifically, how the main created RAG Model GeoBot, which utilises OpenAI's LLM GPT-4, effectively incorporates geographical data from the Gazetteer for Scotland, improving OpenAI's ability to identify local landmarks and Scottish locations. This is supported by GeoBot's superior Cosine Similarity and ROUGE scores compared to default GPT-4 and the created comparative no code RAG model Botsonic. However, there are still weaknesses with this approach as GeoBot performs poorly in response clarity and readability, indicating that further improvement is needed to create the strongest RAG Model methodology. GeoBot also demonstrates the viability of map generation in a RAG Model, although again some improvement could still be made to this. These results highlight the potential application of GeoBot for geospatial applications.

GeoBot can be accessed through this website:

<https://www.geos.ed.ac.uk/dev/geobot/main>. Password is 'GIS2025!'. If this does not connect the website is now offline. The full code can be seen in the Technical Report Appendices.

Introduction

Have you ever asked an Artificial Intelligence (AI) about Scottish geography and it has returned obviously incorrect answers? Would you be pleased if you asked ChatGPT, OpenAI's LLM, about Edinburgh Castle and the AI described it as a small Cathedral in Aberdeen? Most likely not. These response mistakes undoubtedly expose the significant geographical flaws of Large Language Models (LLMs). LLM's are Generative AI models that draw on text datasets to generate responses (CloudFlare, 2022), while Generative AI is systems that learn from data to create content such as text, images and even music (Harvard University, 2025).

Scottish geographical user queries can be complex due to Scotland's distinctive historical and natural landscape (Mckirdy and Crofts, 2010). This can lead to LLM's producing inaccurate responses. For example, OpenAI's LLMs (OpenAI, 2024) and Google's LLM Gemini (Google, 2023) both do not identify that there are three Newbiggings in Angus, villages near Dundee, or provide a precise distance between Arthur's Seat and the Meadows (two greenspaces in Edinburgh). As long as LLM's draw from inadequate datasets this knowledge gap will continue, even beyond Scotland. Because a staggering ten percent of the world uses ChatGPT (Paris, 2025), addressing these LLM knowledge gaps is essential to ensure that all LLM users have access to reliable local geographical information.

LLM's, despite some advantages, also possess other numerous flaws. LLM's are trained on static (unchanging) datasets, so cannot verify the accuracy of training content and struggle to differentiate ambiguous queries (Ribas, 2024). Manvi *et al* (2024) also demonstrate the inherent geographical bias in LLM's, for example how they consistently underestimate the population density of Africa. They also often hallucinate, where a model convincingly generates inaccurate or false information (Farquhur *et al*, 2024).

However, LLMs do demonstrate some strengths in handling general geographical data. Roberts *et al* (2023) tested OpenAI's GPT-4 (OpenAI's latest LLM, standing for Generative Pre-trained Transformer Model) to identify its geographical understanding. GPT-4 correctly identifies locations fitting specific geographical criteria, for example places with both a volcano and a city above a one million population. However, locations are still missed, for example Mount Teide on Tenerife. Improving these capabilities requires more comprehensive datasets that an LLM can draw from.

To remedy this oversight, this research investigates how to improve LLM's for Scottish geographical queries using Retrieval Augmented Generation (RAG), also offering insight into the application of this for the geospatial industry. Two contrasting RAG Models were created to evidence this: GeoBot and a comparative no-code chatbot Botsonic. RAG improves LLM outputs by referencing a provided dataset instead of relying on LLM training data (Amazon Web Services, n.d.d).

GeoBot is a RAG Model enhancing geographical responses from OpenAI GPT-4. It utilises data from the Gazetteer for Scotland, a comprehensive database of Scottish locations (Gittings, n.d), also integrating an interactive map feature. A customizable, no-code chatbot was also created using the Botsonic (n.d) platform, also utilising GPT-4, that integrates external data from Gazetteer for Scotland URL links to provide responses

through a RAG approach. The Botsonic chatbot was created to provide a comparison: can a no-code GPT-4 chatbot, taking less than a day to create, perform just as well as GeoBot? Both RAG Models are aimed at a wide range of adult users, from geographical experts to casual users, although a basic comprehension of geographical locations is assumed.

This paper analyses the current capabilities of AI and Large Language Models. The methodology of the main RAG Model, GeoBot, is explained, alongside the comparative Botsonic RAG Chatbot. Both are then compared to identify the strongest RAG Model methodology, alongside the base OpenAI GPT-4 as a comparison, with suggestions for further improvements and implementations made.

Research Questions

How effectively does drawing upon RAG Model data, instead of original LLM training data, improve LLM Scottish geographical responses?

Does the choice of RAG Model significantly change the quality of LLM Scottish geographical outputs?

Is implementing map generation capabilities into a RAG Model viable?

Aims

To improve LLM Scottish geographical outputs to spatial queries, such as the LLM identifying correct local landmarks and specific locations.

To identify the strongest RAG Model methodology, also considering potential future applications for the geospatial industry.

To create an LLM application that will return a map based on user-inputted spatial information.

Objectives

To create functional RAG Models using OpenAI and a no-code platform, Botsonic.

To compare and critique the outputs from the Botsonic and OpenAI Models, using default base OpenAI GPT-4 as a control, to identify the strongest RAG Model methodology.

To create a Leaflet map generation feature, attaching this to the OpenAI RAG Model.

Artificial Intelligence: Present Capabilities and Future Possibilities

What does it mean for machines to ‘learn’ and ‘think’, characteristics not normally associated with computers, and how does this inform wider understanding of AI’s potential and limitations?

AI is not easily defined. When asked ‘What is Your Definition of AI?’, Gemini (Google, 2023) and OpenAI GPT-4 (OpenAI, 2021) both highlight how AI performs tasks, for example learning and problem solving, which typically require human intelligence. In this context, learning refers to how an AI can improve its response performance by identifying data patterns (SAS, 2010). However, what these LLM definitions fail to recognise is that AI does not possess human intelligence, instead using formulaic statistical pattern recognition drawn from human sources (Morozov, 2023). AI is more accurately defined as systems with apparent intelligence that analyse their environment and use some autonomy to achieve defined goals (HLEG, n.d; Sheikh, Prins and Schrijvers, 2023).

All commercially available AI can be classified as AI Narrow Intelligence. General AI (McKinsey, 2024) and Superintelligent AI (Tsementzis, 2024), which can either equal human intelligence or surpass it, are currently purely theoretical. Narrow AI is only designed for specific tasks, such as language translation (Nancholas, 2023).

The risks of AI Narrow Intelligence are already realised, such as using Raw LLM’s, which have not been trained and could therefore not have ethical frameworks, to design weaponry, alongside exacerbating online misinformation and propaganda (Poulton, 2024). The environmental issues of Narrow AI are also significant. Li *et al* (2023) states that AI infrastructure may soon consume six times more water than Denmark, and Verma and Tan (2024) evidence that ChatGPT (OpenAI’s ChatBot which utilises GPT-4) consumes on average over one 500ml bottle of water per 100-word request. An LLM search also uses ten times more energy than a Google search (Ritchie, 2025), which on a large scale can cause significant environmental harm. AI is essential to produce a geospatial RAG Model. However, the wider issues of this technology must be mitigated, particularly the harmful environmental impact of LLM queries.

Machine Learning and Large Language Models

Machine Learning (ML), a subsection of AI, can learn new information automatically without being explicitly programmed to do so (Sarker, 2021). ML and AI are often interlinked; Brown (2021) states that most advances of AI have involved ML, such as using self-learning algorithms to create predictive models (Neural Technologies, 2024).

Natural Language processing is a further subset of Machine Learning, where machines process language as written and spoken by humans (Brown, 2021). This involves computational linguistics (such as semantic analysis), Machine Learning and Deep Learning (neural networks resembling the human brain) to analyse and decode human language (Amazon Web Services, n.d.c). Generative AI, a part of ML, focuses on creating

new content, such as text and images. One of the common applications of ML, and the focus of this report, is Large Language Models (LLMs).

There are currently limited commercial LLM's designed specifically for geospatial queries, the focus of this research. Chen *et al* (2024) state that LLM's currently struggle to incorporate spatial data structures and tend to hallucinate 'less-used geospatial libraries'. To remedy this, a few spatial LLM's have been created, including Geo-Agent, which uses RAG techniques to produce python code, improving geospatial processing. Yu *et al* (2025) have also created Spatial-RAG, which enhances LLM's spatial reasoning by producing maps based on user queries. However, both these models do not focus on improving the geographical knowledge of LLM responses, instead purely focusing on spatial map generation and geospatial processing. The RAG Models created in this research go beyond map generation to also increase OpenAI's GPT-4 geographical knowledge.

Methodology

RAG Model Justification

This research develops two RAG models, GeoBot (drawing upon OpenAI LLM GPT-4) and a comparative no-code chatbot using the Botsonic platform (also utilising GPT-4). This is instead of creating new LLM models due to the high complexity (Amell, 2023) and cost (\$30,000-\$450,000 (Tatarek, 2023)).

Fine Tuning could also be considered suitable as instead of creating a new LLM, it draws upon LLM pattern recognition capabilities, providing additional training to learn a specific task (Chawre, 2023). Ghosh (2024) states that Fine Tuning is beneficial as the LLM can memorise the provided information. However, this was not chosen due to potential loss of other functionality once trained and costly retraining requirements (Ghosh, 2024).

Directly Inputting data into an LLM, such as OpenAI, was also considered as this can reduce hallucinations and improve responses. Manvi *et al* (2024) evidence how providing LLM's with coordinate data can enable LLM's to make geospatial predictions, improving performance by 70%. However, ultimately this was not chosen due to the risk of unauthorised data disclosure (Bergmann, 2024).

Prompt Engineering, where user inputs are refined to create optimal responses (SAP, 2024), was also briefly examined as it can reduce response bias (Lange and Onyegbula, 2025). However, this does not add new geographical information to the LLM, instead just superficially altering responses, diminishing this method's impact.

Contrastingly, RAG models were selected as using a generic LLM in a RAG Model, which is acting on a known and reliable dataset (Amazon Web Services, n.d.d), allows the conversational capabilities of the underlying LLM to be preserved while also being more cost-effective due to not needing extensive retraining (Shone, 2025). Figure 1 shows how RAG models retrieve database data to respond to user queries.

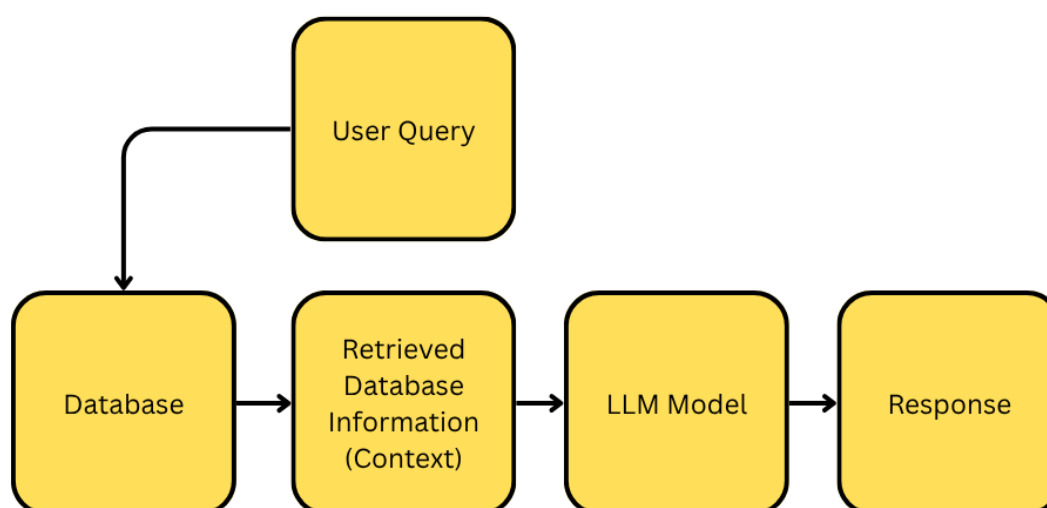


Figure 1: Logical RAG Architecture

However, there are still disadvantages to this approach. RAG models can struggle with returning the most relevant information (Besen, 2024), instead just returning data that has the exact text to the user query. Nevertheless, this RAG methodology was deemed suitable as this can be mitigated by utilising techniques such as Similarity Search, which identifies information most like the user query (Ozkaya, 2024).

Geographical Concepts

It was essential that the two RAG Models were grounded in geographical concepts (Table 1). These provide a geographical framework to inform the RAG Model capabilities, shown in Table 2.

Table 1: Relevant Geographical Concepts

Concept	Description
Space	How features (for example services, physical features) are arranged on the earth's surface. This includes concepts such as location (coordinates), pattern, distribution, interaction and distance (Geographical Association, n.d.). This can be thought of as empirical (directly measuring space), flow space (how pathways are bonded together through interactions), space (how we view locations through images) and place (what gives a space meaning or cultural identity (Cliffords <i>et al</i> 2008, p.87-90).
Place	The origin and physical/social characteristics of a location (National Geographic, n.d.). Cresswell (2015) also defines place as a location, community, landscape and a way of understanding the world.
Environment	The interplay of physical and human geography, ecosystems, environmental change, resources and sustainability, in a variety of contexts and scales.' (Geographical Association, n.d.).
Earth Systems	The physical processes of the earth, for example physical landscape changes (Geographical Association, n.d.). The interacting 'physical, chemical and biological processes' (International Geosphere-Biosphere Programme, n.d.)
Time	Gives the context of past, present and future, for example change and stability (Geographical Association, n.d.)
Scale	Viewing places at all levels from local to global to analyse the place itself and relationships between (Geographical Association; Cliffords <i>et al</i> , 2008).
Diversity	Looking at the similarities between people, places, environment and cultures (Geographical Association, n.d.).
Interconnection	How places are linked together and how this affects each location (Geographical Association, n.d.)
Interpretation	How the world is influenced by changing narratives and viewpoints (Geographical Association, n.d.).

Table 2: RAG Model Geographical Concept Application

RAG Model Capabilities	Description
General understanding of Scottish Places (Place, Environment)	Correctly identify place names and locations.
Concept of Nearness and Inferencing from Coordinates (Space)	Calculate precise distances between places.
Concept of Associations (Interconnection)	Identify features about places and then associate them with each other.
Concept of Clustering (Interconnection, Space)	Recognise when places are clustered together e.g. a hot spot of historical landmarks.
Concept of Overlapping and Ambiguity (Space, Interpretation)	Recognise when boundaries between places are ambiguous or the answer is unclear.
Concept of Time	Recognise how geography can change over time.
Map Generation	Can return any inputted coordinates (inferred or stated) as a leaflet map (Place, Space).

RAG Model Input Questions

The input questions (Table 3) must also be carefully considered. If the RAG model does not correctly understand the input question due to misspellings or vagueness this can have a detrimental impact on response quality. To reduce the dataset size, these questions revolve around notable greenspaces in Edinburgh (except for Newbigging in Angus). Zero-Shot Prompt engineering (where you provide no examples in your query) (Microsoft, 2024) was chosen for the following baseline queries. This was considered the clearest and most understandable approach for this research.

Table 3: RAG Model Result Test Questions

Geographical Concepts	Test Questions
General understanding of Scottish Places (Place, Environment)	Question One: What are the geographical features of greenspaces in Edinburgh?
Concept of Nearness and Inferencing from Coordinates (Space)	Question Two: How far is it between Arthur's Seat and the Meadows in Edinburgh? Does this distance vary depending on the mode of transport used?
Concept of Associations (Interconnection)	Question Three: Which greenspaces in Edinburgh are connected with famous figures, and what are their historical links?
Concept of Clustering (Interconnection, Space)	Question Four: Where are the major clusters of greenspaces located in Edinburgh? What common features or characteristics do these greenspaces have?
Concept of Overlapping and Ambiguity (Space, Interpretation)	Question Five: What is the precise location of Newbigging in Angus? Can you describe any landmarks or features nearby?
Concept of Time	Question Six: How was Arthur's Seat formed? What significant geological events have impacted it over time?"

Choice of RAG Model Platform

The RAG Model platform choice was based on the following criteria: methodology complexity, pricing, customisation capability, environmental considerations and scalability. For the main RAG Model, Cloud Service Providers Google Cloud Platform (n.d) (Google, 2023) and Amazon Web Services (n.d.d) were considered. Both provide cloud-based pre-built RAG Methodologies, an example of which is shown in Figure 2.

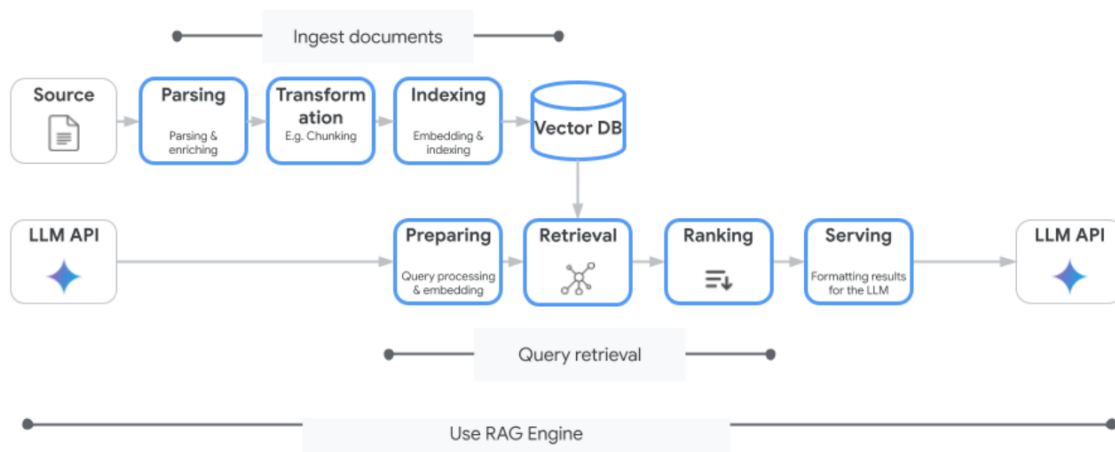


Figure 2: Google's Pre-Built RAG Model Methodology (Google, 2025)

Alternatively, OpenAI provides just an LLM API Key, allowing integration in customised programming applications instead of using a Cloud Provider (OpenAI, 2024). An API (Application Programming Interface) key is a code to authenticate a user (Fortinet, 2023). All options operate on a pay-as-you-go model with pricing structures based on usage, with free trials available.

OpenAI was chosen instead of AWS and GCP as it enables complete customisation as the user constructs their own surrounding programming application, whereas GCP and AWS are partially constrained by their default RAG pipelines, although customised programming attachments can be added.

GeoBot: OpenAI RAG Model Summarised Methodology

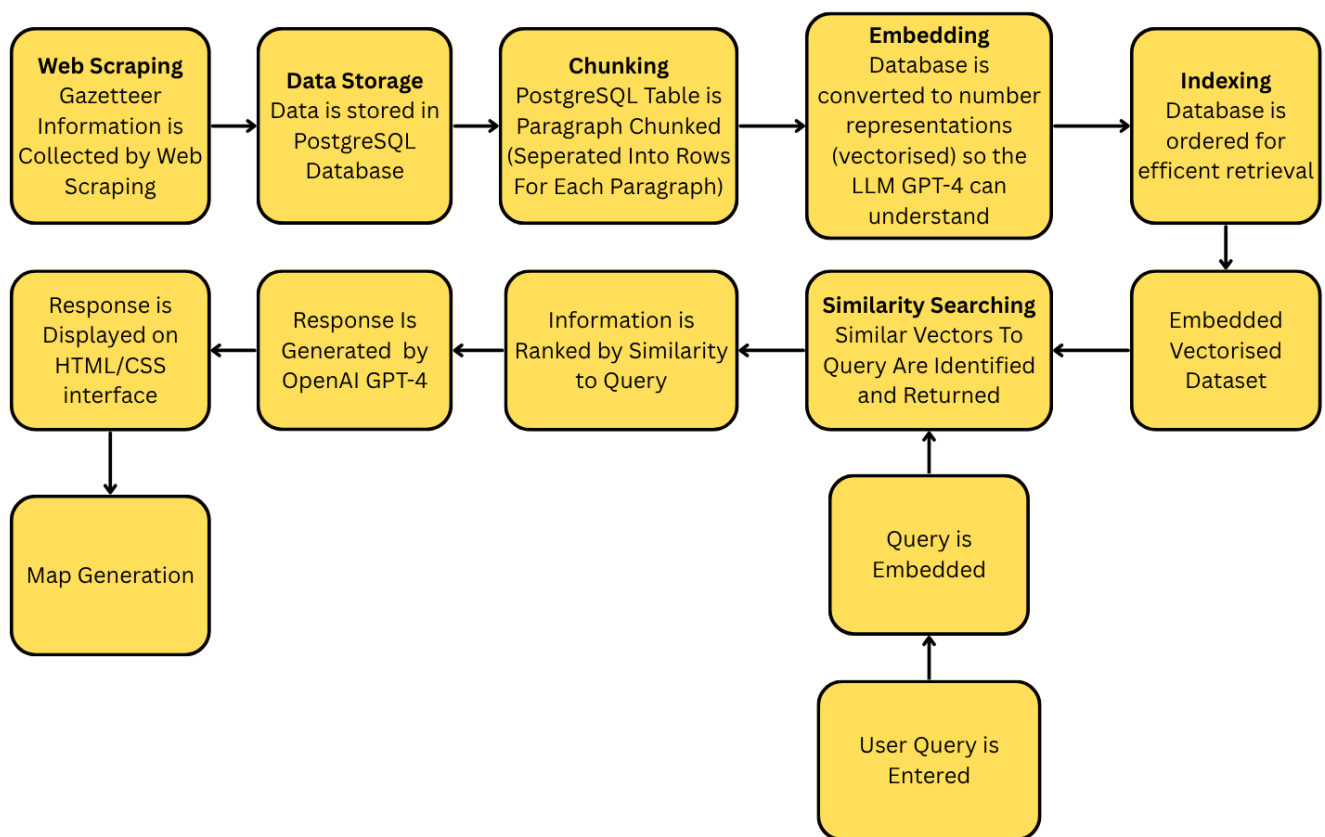


Figure 3: GeoBot RAG Model Workflow

Figure 3 shows the main GeoBot components. The Gazetteer for Scotland data is collected, stored and processed. The most similar information to the user query is returned then a response is created by OpenAI GPT-4 and presented on the GeoBot HTML website interface.

OpenAI is an AI research and deployment company (OpenAI, 2024) who have released a variety of LLM models. The latest, GPT-4, is a large closed-source generative pre-trained transformer model, which can intake and generate both text and images (multimodal) (OpenAI, 2021). This RAG Model methodology (Figure 3) utilises ELM the Edinburgh (access to) Language Models platform (University of Edinburgh, 2024), which in turn uses OpenAI GPT-4. This was chosen as it allows innovation on top of ELM through an Application Programming Interface whilst prioritising data privacy. This GeoBot methodology does not specifically require ELM, but it was used in this case to access the OpenAI GPT-4 model. Table 4 shows a summary of the data processing and retrieval process.

Table 4: Data Processing and Retrieval Methodology

Methodology Process	Description	Justification and Alternatives
Web Scraping	Retrieves information from a webpage and stores in a database (National Library of Medicine, 2022).	Selenium (n.d) was chosen over BeautifulSoup (n.d) for its ability to navigate website subsections, essential for accessing two sub-webpages per Gazetteer location.
Data Storage	The scraped Gazetteer for Scotland Data is stored in a database.	The open-source PostgreSQL database (Oracle, 2014) was selected for its ease of use and cost-effectiveness. The closed-source Gazetteer Oracle database was not used due to high embedding costs.
Chunking PostgreSQL Data	Chunking splits large datasets into smaller pieces of text (Kutumbe, 2024).	Paragraph chunking maintained the Gazetteer for Scotland sentence structure. Alternatives sentence based chunking and fixed chunking (based on a fixed number of words) would have harmed the text meaning.
Embedding Data	Embedding converts text into numbers, allowing LLM's to understand the data (Hernandez, 2023).	The model 'text-embedding-ada-002' was chosen for its simplicity and low cost, priced at 99.8% lower than OpenAI's alternative model Davinci (GPTHub, 2022).
Indexing Data	Indexing clusters the database, allowing the RAG Model to retrieve the most similar query information (Yashwanth, 2025).	IVFFlat (Inverted File with Flat Compression) (Bayer, 2025) was chosen as despite being less accurate than HNSW (Hierarchical Navigable Small World), IVFFlat was chosen due to its speed and low memory requirements.
Similarity Searching	Similarity Search finds similar vectors (number representations of words) to a query vector (HopsWorks, 2023).	Euclidean distance measures the straight-line distance between two vectors (Schwaber-Cohen, 2023) and can process both the magnitude and direction of vectors (essential for coordinate processing) (Escape Force, 2025), unlike cosine similarity.

RAG Model Interface

Once the relevant information has been retrieved it is sent to the OpenAI API which then creates a response. This is displayed on a webpage using HTML, CSS and JavaScript. The modern and minimal interface design was inspired by OpenAI GPT-4 (OpenAI, 2021) and ELM (University of Edinburgh, 2024), shown in Figure 4.

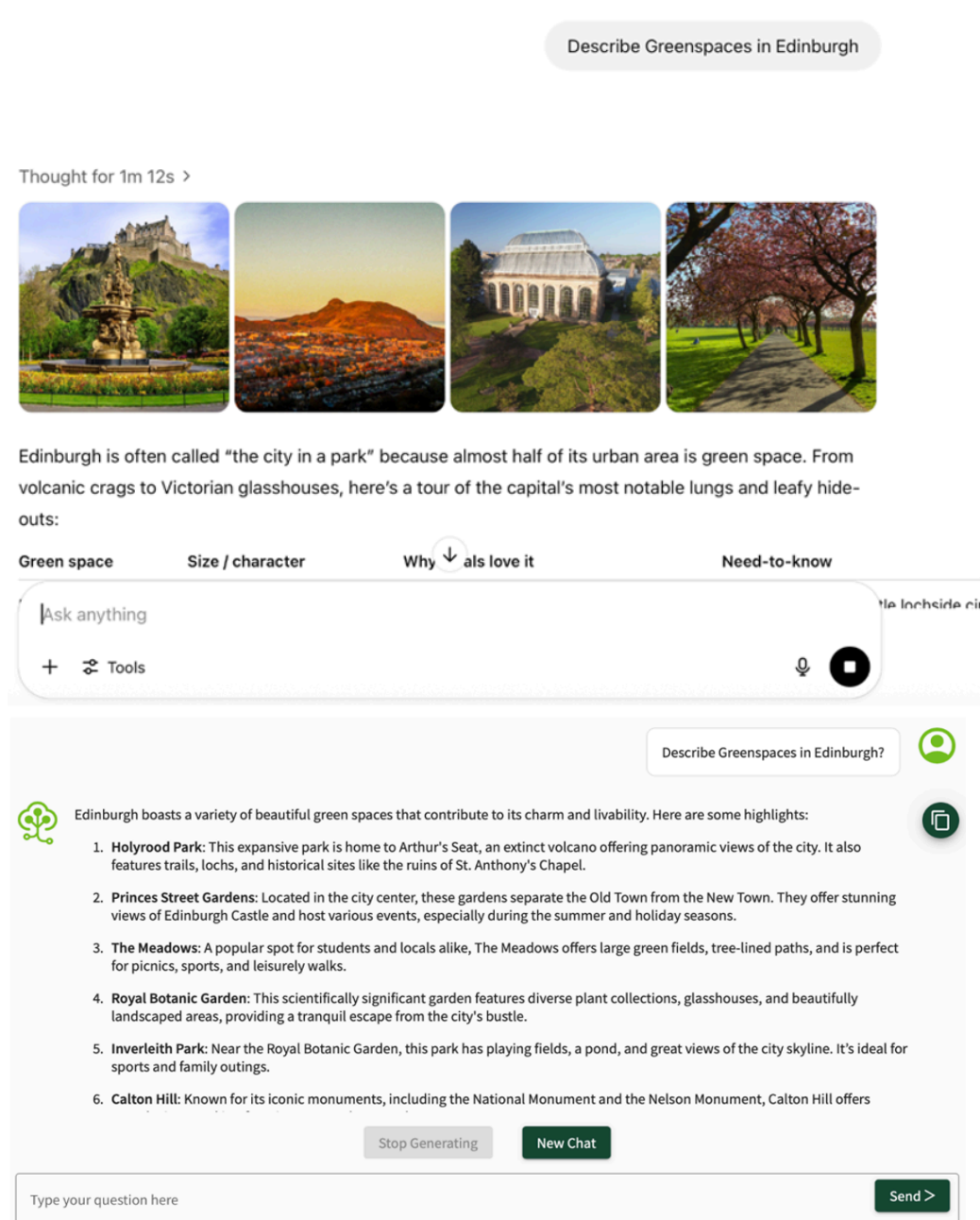


Figure 4: OpenAI GPT-4 Interface (Top) (OpenAI, 2021) and ELM (University of Edinburgh, 2024) Interface (Bottom)

Map Creation

If there are coordinates in the GeoBot response a Leaflet map of the referenced places can be created. The Python library Folium visualises data that has been manipulated in Python on an interactive Leaflet map (Leaflet, 2019). This was chosen over static map creation using the Python library Cartopy because of Leaflet's high interactivity.

No-Code Alternative RAG Chatbot

A no-code RAG Chatbot was used as a direct comparison to GeoBot. These enable RAG Models to be created by users without any prior programming experience. Botsonic (n.d) uses OpenAI GPT-4 to generate chatbot responses. This was chosen because of its simple user interface and diversity of allowed dataset inputs (PDF's, website links, Excel Spreadsheets) (Woyera, 2024). Gazetteer for Scotland website links were entered to match the Greenspace data in the PostgreSQL table. Despite both OpenAI and Botsonic both using GPT-4, Botsonic was chosen to compare the user experience and methodology of each RAG Model, rather than the LLM itself.

RAG Model Result Methodology

The result methodology drew upon known LLM evaluation metrics (Table 5). McGrath and Jonker (2024) state that LLM evaluation is essential as it can identify model performance and further development opportunities. The two RAG models (GeoBot and Botsonic) were compared, as well as base OpenAI GPT-4 as a baseline comparison.

Table 5: RAG Model Result Metrics

Metric	Source
Semantic Similarity	Google Cloud (n.d); RAGAS (n.d)
Response Accuracy	Zhang (2024); Google Cloud (n.d)
Response Clarity	Amin <i>et al</i> (2024)
Response Consistency	Ni (2024)
Task Success Rate	Zhang (2024)
User Interaction and Satisfaction	Quinn and Lecker (2024)

Semantic Similarity

Semantic Similarity measures the similarity between the generated response and the database source information (RAGAS, n.d). The metric Cosine Similarity was chosen which analyses the similarity between the orientation of two vectors (Karabiber, n.d). The higher the score (between 0-1), the higher the response similarity. This was chosen as it does not consider each vector's magnitude (Nagella, 2019), making it suitable for LLM responses where text length varies. Alternatively, Euclidean Distance measures the straight-line distance between vectors (Crossman-Smith, 2025) and was not chosen as it does consider magnitude, meaning it can be heavily affected by text length.

Response Accuracy

ROUGE Score evaluates the recall and precision of the response compared to the reference text (Herroros, Veasay and Papaoikonomou, 2023), instead of purely calculating the similarity. The higher the score (between 0-1), the higher the response accuracy. Bleu score, which using matching sequences of words to calculate response

precision (Santhosh, 2023) was also considered, however was ultimately discarded due to its emphasis on exact sentence matches, which is not preferable for AI generated text due to LLM's changing the syntax of the database information.

Clarity

The response clarity is essential as it evidences the RAG Model accessibility. This metric was measured through the Flesch Reading Ease Test (Readable, n.d) as it considers both sentence length and complexity, unlike the Automated Readability Index (ARI) (Readable, n.d), with the higher the result signalling greater readability (Microsoft, n.d.a).

Consistency

Jaccard Similarity calculates the similarity between two sets of text by calculating the word intersection ratio (Karabiber, n.d). This was chosen as it can determine similarity between text data of different sizes (Jadeja, 2022), unlike Bleu score, as the RAG Model response size differs from that of the database information.

User Satisfaction and Task Failure Rate

Grigoryan (n.d) states that user feedback surveys can validate product ideas and find areas of improvement. Therefore, a Microsoft (n.d.b) user feedback survey was used to record response quality and relevance, as well as Task Failure Rate, in an uncontrolled environment. Task Failure Rate is the frequency in which the chatbot unsuccessfully answers a query, essential when measuring a RAG Model's quality.

Results

GeoBot and Botsonic, alongside base OpenAI GPT-4 as a comparison, were analysed using six test questions (Table Three), focusing on their effectiveness as RAG models for geographical queries. To access the extent to which RAG Models remedy the Scottish geographical gaps in LLM's, the evaluation metrics response accuracy, similarity, clarity, consistency and user satisfaction were chosen.

GeoBot Interface

Figures 5 and 6 show the final GeoBot interface and a subsequent map generation example.

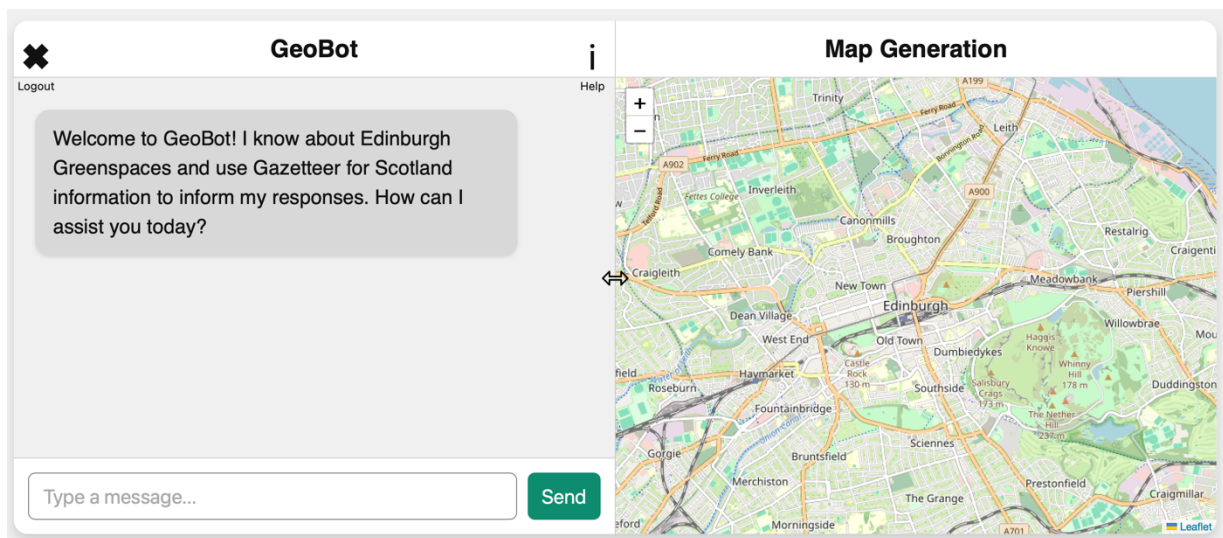


Figure 5: GeoBot Interface

GeoBot Map Generation

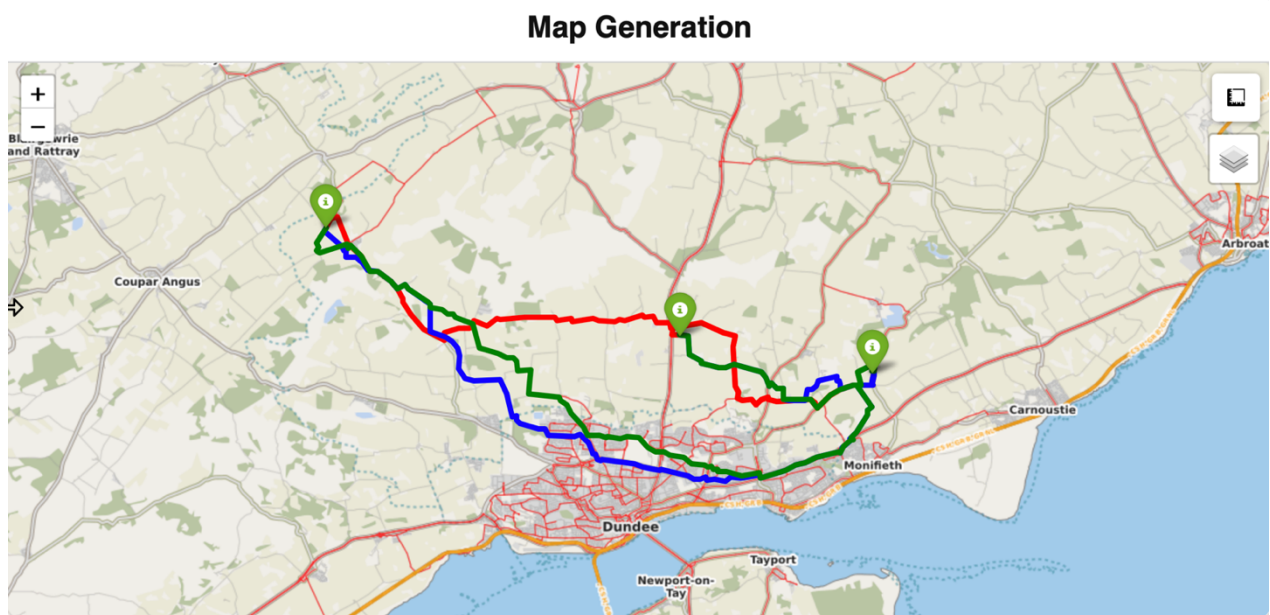


Figure 6: GeoBot Leaflet Map of the Three Newbiggings in Angus

Figure 7: Example Tested ChatBot Responses

RAG Model Responses for Question Five: What is the precise location of Newbigging in Angus? Can you describe any landmarks or features nearby?

GeoBot

Newbigging in Angus includes multiple locations with the same name, each distinguished by different surroundings and landmarks:

1. Newbigging near Newtyle

Description: This small hamlet is located close to the western boundary of Angus, near Newtyle. It is notable for its proximity to other rural communities such as Ardler and Meigle in Perth and Kinross.

Coordinates for Newbigging, Angus: Latitude: 56.5645, Longitude: -3.1617. Follow this [link](#) for more information about Newbigging, Angus.

2. Newbigging near Monikie

Description: This village in southeast Angus expanded in the 1950s with the construction of public housing. Nearby, stone is quarried, and historical sites like the Carlungie and Ardestie souterrains are located to the east and south, respectively.

Coordinates for Newbigging, Angus: Latitude: 56.5142, Longitude: -2.8189. Follow this [link](#) for more information about Newbigging, Angus.

3. Newbigging near Tealing

Description: Positioned in south Angus, this hamlet is immediately east of the A90 trunk road, close to Tealing and north of Dundee. Its location near main transport routes highlights its connectivity to larger urban areas.

Coordinates for Newbigging, Angus: Latitude: 56.5268, Longitude: -2.9395. Follow this [link](#) for more information about Newbigging, Angus.

Each location provides unique insights into the rural life of Angus with its distinctive landmarks and cultural heritage. Please specify if you need information about one particular Newbigging or further details on the nearby features.

Alternative Chatbot (Botsonic)

Newbigging is a small village located in southeast Angus. It lies approximately 1¼ miles (2 km) south of Monikie and about 2 miles (3 km) east-southeast of Wellbank. Additionally, it is situated immediately to the east of the A90 trunk road, about half a mile (1 km) southeast of Tealing and 4 miles (6.5 km) north of Dundee.

In terms of notable features, Newbigging has a history tied to public housing development in the 1950s. Nearby, stone is quarried, and there are ancient souterrains located to the east at Carlungie and to the south at Ardestie. These features contribute to the village's unique character and historical significance.

Base OpenAI GPT-4

Newbigging is a small village located in Angus, Scotland, near Monifieth. It lies approximately 5 miles northeast of Dundee.

Notable landmarks and features in or around Newbigging include:

Monikie Country Park: Situated close by, it offers outdoor activities like walking and water sports.

Barry Mill: A historical water-powered mill located a few miles east of Newbigging.

Panmure Testimonial: A prominent stone monument visible from the village, located in the nearby parklands. The area is primarily rural, with scenic landscapes and farmland surrounding the village.

Summary

GeoBot correctly identifies that there are three Newbiggings in Angus and produces detailed geographical coordinate data for each location. Botsonic does provide a more geographically detailed response than GPT-4, but both Botsonic and ELM still fail to realise that there is more than one Newbigging in Angus.

Response Accuracy and Similarity

Combined Cosine Similarity and ROUGE Scores

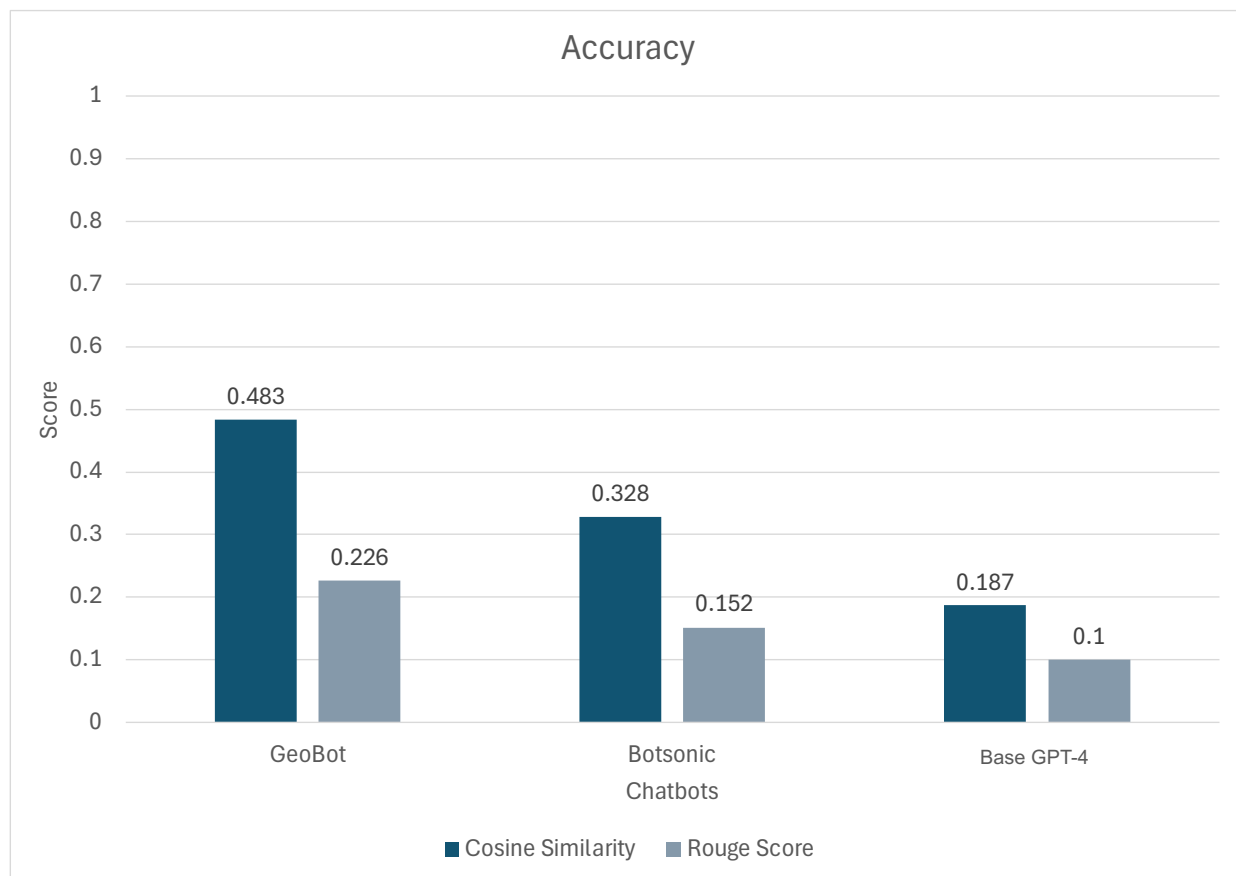


Figure 8: Cosine Similarity and ROUGE Scores of Each Chatbot. Scores Range From 0: No Similarity or Accuracy To 1: Complete Similarity or Accuracy

Table 6: Cosine Similarity and ROUGE Score Data Table

Chatbot	Average Cosine Similarity	Average F1 ROUGE Score
GeoBot	0.483	0.226
Botsonic	0.328	0.152
Base GPT-4	0.187	0.1

Figure 8 shows the overall Cosine Similarity (CS) and ROUGE score for each chatbot. GeoBot’s CS score of 0.483 indicates a moderate similarity between the database information and the GeoBot response (Datacamp, 2024). When compared with the default OpenAI GPT-4, which the GeoBot RAG Model builds upon, this improvement is particularly stark. Base GPT-4 only scores 0.187, evidencing extremely low similarity (Datacamp, 2024). Although not to the same extent as GeoBot, the Botsonic RAG Model also shows an 75% CS improvement when compared with Base GPT-4.

This trend is also shown in the ROUGE scores (RS), albeit to a much lesser extent. GeoBot scores the highest at 0.226, compared to the low Botsonic and base GPT-4 scores of 0.152 and 0.1. However, the RS scores all indicate a low overall response accuracy (Santhosh, 2023).

Cosine Similarity Scores For Each Test Question

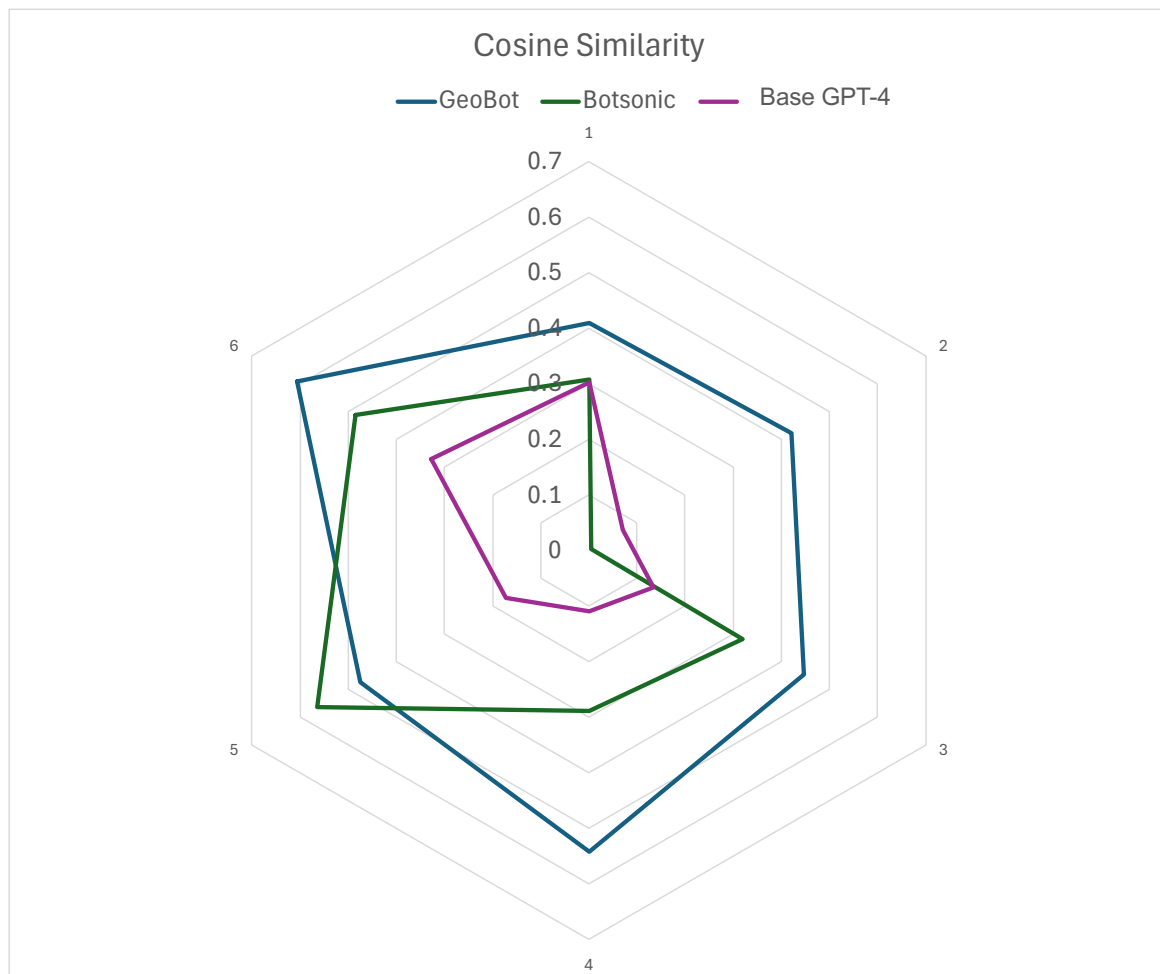


Figure 9: Cosine Similarity Scores for Each Question. Scores Range From 0: No Similarity To 1: Complete Similarity

Table 7: Cosine Similarity Data Table

Chatbot	Q1	Q2	Q3	Q4	Q5	Q6
GeoBot	0.408	0.422	0.447	0.543	0.475	0.606
Botsonic	0.307	0.004	0.319	0.289	0.565	0.485
Base GPT-4	0.301	0.071	0.135	0.111	0.173	0.328

Figure 9 focuses on the CS score for each test question. GeoBot performed the strongest consistently, indicating a high similarity between the response and the database information, except for question five (concept of overlapping and ambiguity) where Botsonic scored the highest. Question six (concept of time) achieved the highest CS scores (0.606 and 0.485 respectively). Contrastingly, question two (concept of nearness and inference from coordinates) scored the lowest overall.

ROUGE Scores For Each Test Question

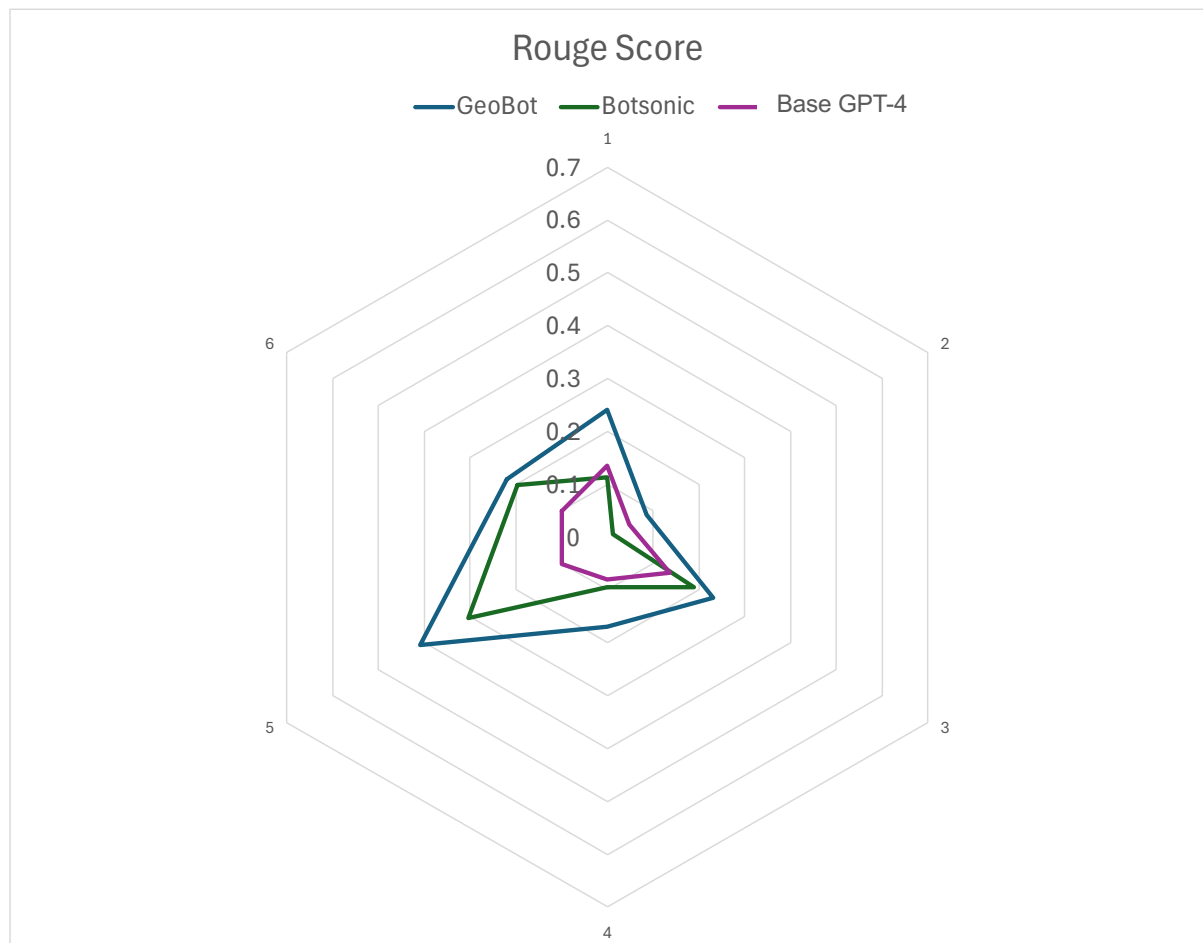


Figure 10: ROUGE Scores for Each Question. Scores Range From 0: No Accuracy To 1: Complete Accuracy

Table 8: ROUGE Score Data Table

Chatbot	Q1	Q2	Q3	Q4	Q5	Q6
GeoBot	0.2411	0.0848	0.231	0.1703	0.409	0.22
Botsonic	0.114	0.012	0.189	0.0953	0.305	0.196
Base GPT-4	0.135	0.048	0.135	0.08	0.1	0.1

Figure 10 shows the ROUGE scores for each of the test questions. As with the CS Score, GeoBot scored the highest for all six test questions. Question five (concept of overlapping and ambiguity) had the highest scores for both GeoBot and Botsonic. Contrastingly, question two (concept of nearness and inference from coordinates) scored the lowest.

Response Clarity

Flesch Reading Score

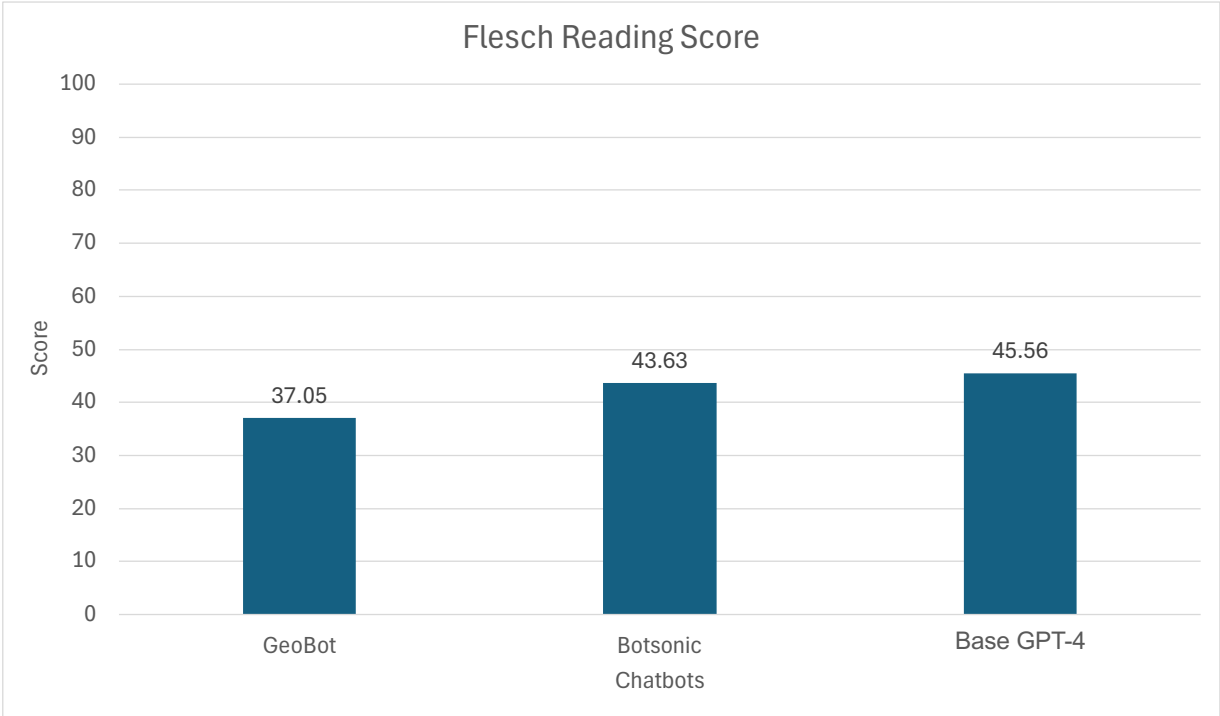


Figure 11: Flesch Reading Ease Score for Each Chatbot

Table 9: Flesch Reading Ease Score Data Table

Chatbot	Average Flesch Reading Ease Score
GeoBot	37.05
Botsonic	43.63
Base GPT-4	45.56

Table 10: Flesch Reading Ease Scoring Table with Chatbots score range highlighted (Readable, 2024)

Score	Description
90-100	Very Easy
80-89	Easy
70-79	Fairly Easy
60-69	Standard
50-59	Fairly Difficult
30-49	Difficult
0-29	Very Confusing

Figure 11 shows the Flesch Reading Ease Score for each chatbot. The base OpenAI GPT-4 performed the strongest. However, all chatbots scored between 37 and 46, meaning that their responses were difficult to read (Table 10).

Response Consistency

Jaccard Similarity

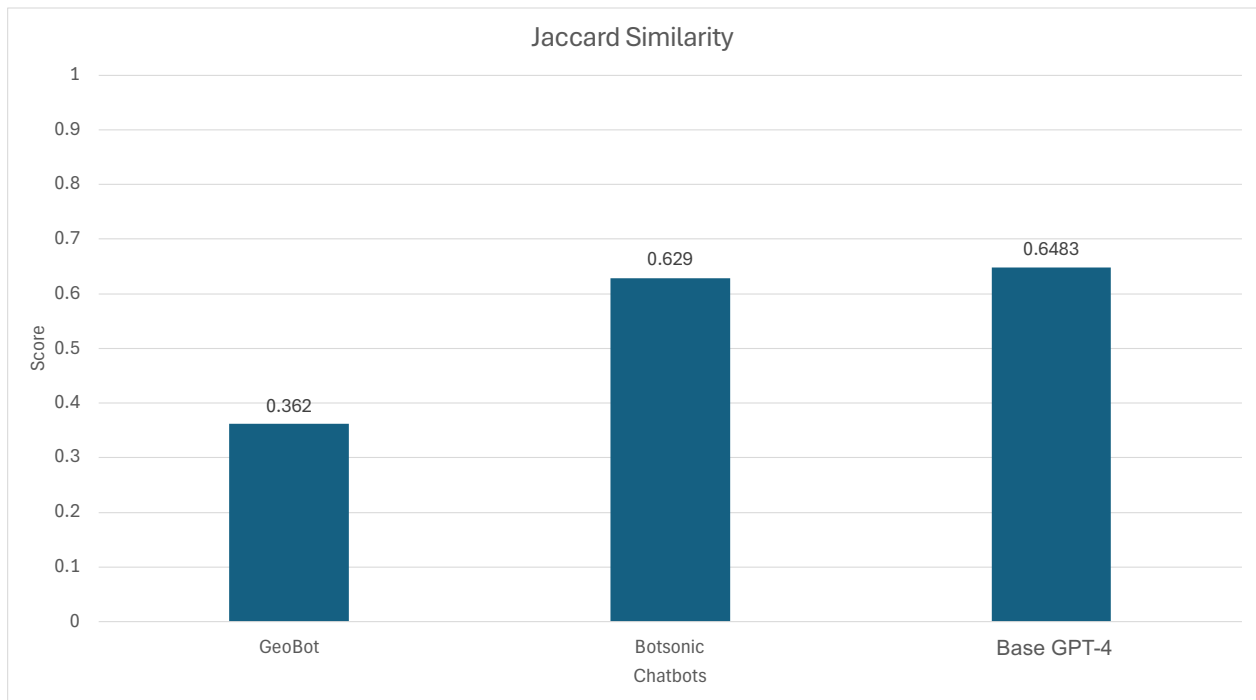


Figure 12: Jaccard Similarity Scores for Each Question. Scores Range From 0: No Consistency Between Responses To 1: Complete Consistency Between Responses

Table 11: Jaccard Similarity Data Table

Chatbot	Average Jaccard Similarity
GeoBot	0.362
Botsonic	0.629
Base GPT-4	0.6483

Figure 12 shows the overall response consistency of each chatbot. Base GPT-4 and Botsonic performed the strongest, both scoring a 0.6 similarity level, indicating there was moderately low variation between responses to the same query (Zhou, 2024). Contrastingly, GeoBot scored the lowest (0.362), showing a high variation between responses.

User Feedback Survey Results for User Satisfaction and Task Failure Rate

Overall User Rating

Table 12: Overall User Rating Feedback Table

Category	GeoBot	Botsonic
Strongest Performance	9 (100% preferred GeoBot)	0 (0% Preferred Botsonic)
Accurate Responses	9	0
Ease of Use	0	9
Overall Preference	9	0

Table 12 shows a user preference towards GeoBot when considering performance, accuracy and overall preference. However, for ease of use all respondents (9) preferred Botsonic.

RAG Model Response Failure Rate

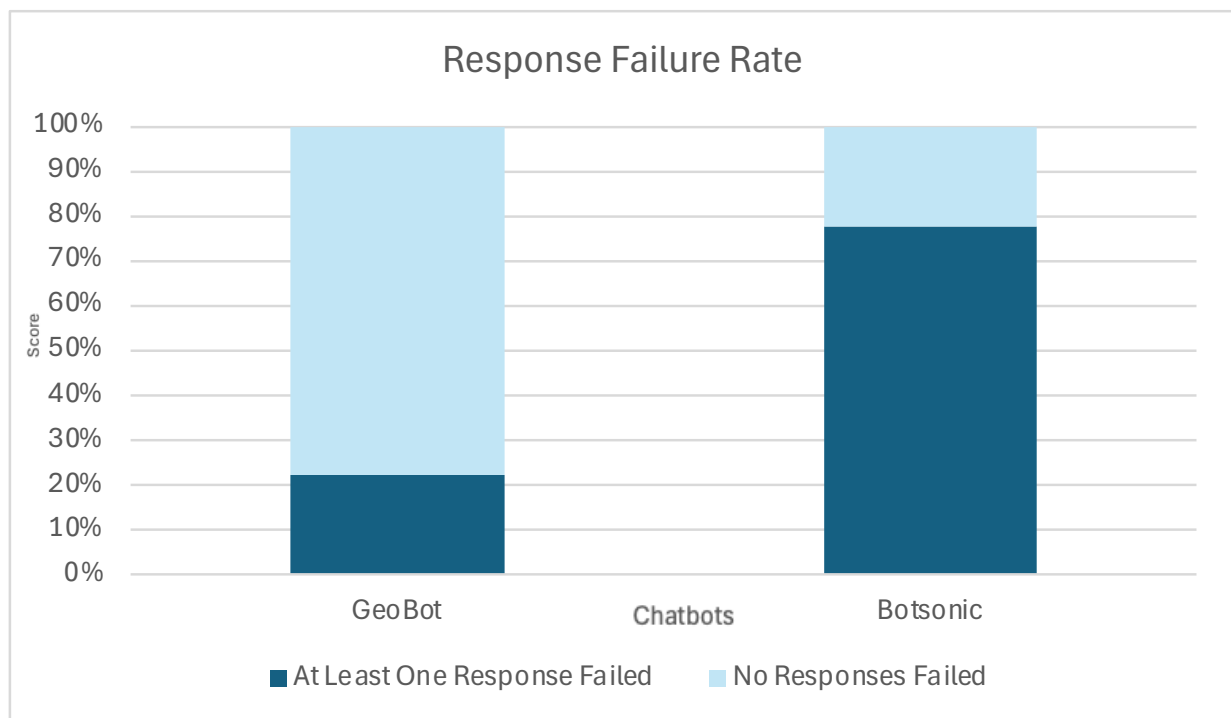


Figure 13: Response (Task) Failure Rate

Table 13: Task Failure Rate Data Table

Response	GeoBot	Botsonic
Yes (at least one response did fail)	22.22% (2/9)	77.78% (7/9)
No (responses did not fail)	77.78% (7/9)	22.22% (2/9)

Table 13 evidences the rate at which Botsonic and GeoBot did not successfully answer a query. The task failure rate, while not being unanimous, does show a significantly higher proportion of unsuccessful Botsonic responses (77.78%) than GeoBot (22.22%).

Map Generation User Feedback Responses

Table 14: Generated Map Accuracy Rate

Test Question	Response Amount
1-4	100% responded that maps were accurate (9/9)
5-6	88.89% responded that maps were accurate (8/9)

Table 15: Map Clarity and Accessibility

Criteria	Response Amount
Extremely	100% responded that maps were extremely accurate (9/9)

Tables 14 and 15 showcase the accuracy and accessibility of the generated GeoBot maps. For questions 1-4, 100% of the maps generated were accurate, however this reduces by 11.11% for questions 5-6, indicating some inaccuracies. All respondents rated the maps extremely accessible and clear to navigate.

Discussion

This discussion evaluates the use of GeoBot and Botsonic in enhancing LLM Scottish geographical responses. Specifically, how drawing upon the Gazetteer for Scotland dataset over the original OpenAI GPT-4 database improves response accuracy, the effect of RAG model choice on response quality and the validity of integrating map generation capabilities. The RAG Models performance is assessed and critiqued, with further improvements and applications presented.

Effectiveness of RAG Model Choice and Data on Enhancing LLM Scottish Geographical Responses

Response Accuracy and Similarity

The strong Cosine Similarity (CS) scores (Figures 8 and 9) show that GeoBot improves OpenAI GPT-4 Scottish geographical responses through its high accuracy, returning responses that closely match the Gazetteer for Scotland information. For example, it accurately identifies that there are three Newbiggings in Angus (Figure 7), unlike base OpenAI GPT-4 and Botsonic. The Botsonic RAG Model also shows a CS improvement when compared with default OpenAI GPT-4 (Figure 8). While still 32% lower than GeoBot, this does identify that a no-code RAG Model can also improve geographical LLM responses.

While base OpenAI GPT-4 was likely not trained on the Gazetteer for Scotland information, meaning that the similarity was bound to be lower (a weakness of this methodology), the low CS scores show that base GPT-4 does not naturally include the detailed Scottish geographical information found in The Gazetteer for Scotland and GeoBot, and further highlights how RAG Models can improve Scottish geographical responses.

GeoBot is still shown to be the most accurate through its relatively high ROUGE scores (RS), however these results (Figures 8 and 10) indicate a low response accuracy for all the chatbots (GeoBot, Botsonic and base GPT-4). This shows that further improvement is still needed to improve the RAG Model responses. Perhaps an alternative similarity search method, such as the dot method which considers both the angle and magnitude of the vectors (Crettaz, 2025), instead of cosine similarity, would improve this RS score.

The CS and ROUGE results also indicate moderate incorporation of Scottish geographical concepts in GeoBot and Botsonic. The concept of time was the most successfully integrated and the ambiguity concept also scored well (Figures 9 and 10). However, the question on nearness and coordinates performed poorly, showing that GeoBot and Botsonic sometimes failed to effectively use the Gazetteer for Scotland data. Inclusion of additional datasets, such as coordinate data, would likely improve GeoBot's and Botsonic's responses for the lower scoring geographical concepts. Testing both RAG models with a wealth of geographical concept questions would also improve the strength of this argument, as this analysis was limited to only one question for each geographical concept.

These CS and RS results do show a definite improvement in accuracy and similarity when utilising a RAG model, especially GeoBot, evidencing the success of the RAG models in

incorporating Scottish geographical information, although further improvement is needed.

Response Clarity

Due to the wide range of adult users, a high reading comprehension score is essential. However, both GeoBot and Botsonic score poorly on readability, particularly affecting GeoBot's high accuracy with a score of 37.05 (Figure 11). Simplifying GeoBot's tone through altering GeoBot's system role could improve this, but it must be balanced to maintain response quality. However, this issue is common as base GPT-4 also performs poorly. Additional testing with alternative response clarity tests, such as the Automated Readability Index which uses characters and sentence length to predict readability (Scott, 2025), would also indicate whether this low score is an issue with the tested chatbots or with the utilised response clarity test.

Response Consistency

Figure 12 highlights GeoBot's failure in response consistency as base GPT-4 is 79% more consistent. This inconsistency leads to more hallucinations (incorrect information) in GeoBot's responses. Alternatively, Botsonic scores 0.6, evidencing a strong response consistency and showing that a no-code RAG Model can sometimes outperform GeoBot. Further testing with additional evaluation metrics would identify GeoBot's consistency failure in more detail, enabling technical improvements such as improved role prompting.

User Satisfaction and Task Failure Rate

User satisfaction favours GeoBot (Table 12). It is deemed the strongest RAG model by users, with a much lower task fail rate (Figure 13) than Botsonic. However, GeoBot still failed 22.22% of the queries. This shows that further refinement is still needed, perhaps by increasing the user capacity of GeoBot. The small user feedback survey size (only 9 respondents) also significantly limits the strength of the results, and more respondents would have been greatly beneficial to increase the validity of this results and make a definite conclusion.

User Interface

The GeoBot user interface (Figure 5 and 6) incorporates the stylistic elements from OpenAI and ELM seen in Figure 4. However, user respondents show preference for Botsonic for ease of use, indicating that further improvements to GeoBot's user interface are needed. To improve this, the map generation box could only appear when suitable coordinates are identified in the response. Currently, GeoBot can only be viewed on a computer which restricts the number of users. To remedy this, the CSS website styling could be altered to allow GeoBot to be viewed on a variety of device sizes. A tutorial could also be added to highlight GeoBot's features and increase user understanding.

While GeoBot does have a high user satisfaction, it nevertheless scores poorly for readability, consistency, and user interface, suggesting improvements would be beneficial in response tone and accessibility. Contrastingly, the no-code chatbot Botsonic scores well for response consistency and user interface simplicity. This shows that a no-code chatbot may be preferable for use cases where clarity and ease of use are more beneficial, such as simple location-based recommendations, however it still has a higher task failure rate compared to GeoBot and so caution must be taken.

The Viability of Integrating Map Generation Capabilities into a RAG Model

GeoBot proves that integrating a map generation feature into a RAG Model is viable. Figure 6 shows a successful map generation, and all survey respondents reported high map generation accuracy and accessibility (Table 14 and Table 15). However, this could still be improved beyond basic coordinate and route display. The map feature could process location coordinates from The Office of National Statistics (n.d) to display census data. Zhang et al (2024) also designed GeoGPT, which can automate and present GIS spatial analysis. With improvements made to the GeoBot mapping system, such as the inclusion of Open Street Map datasets and the incorporation of GeoJson files to show the polygon shape of greenspaces instead of just a marker, GeoBot could similarly process and display geospatial data for more complex spatial analysis.

GeoBot Geospatial Industry Application

These results show that RAG Models, especially GeoBot, are a viable approach to increase Scottish geographical knowledge in LLM's. GeoBot's ability to provide geographical knowledge while reducing LLM hallucinations (Shone, 2025) could be highly advantageous to the geospatial industry. While GeoBot is specific to Scottish geography, the overall methodology could be applied to other geospatial use cases for minimal cost. If the database information is altered, GeoBot could potentially be used to assist with urban planning and environmental monitoring. It could quickly provide relevant information and coordinates for strong map generation.

A mobile app would expand GeoBot's reach as it is currently limited to a website on a computer (93% of UK households have a mobile phone (Office for National Statistics, 2022)). This could be similar to the ArcGIS Field Map App, which incorporates map viewing, data collection and real-time location sharing (Esri, n.d), and Google Maps (n.d), a web-mapping platform. Figure 14 evidences an example of the GeoBot mobile app interface.

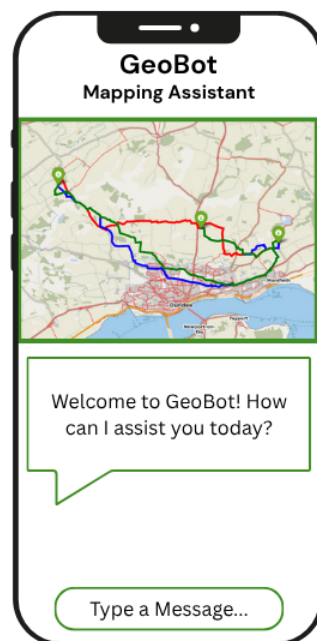


Figure 14: GeoBot Mobile App Interface Concept

GeoBot would initially require the user to input their geospatial data, perhaps through a secure website that is linked with the app to allow large file transfers. This would then be used in GeoBot's responses and map generation. For example, an urban planner could input their geospatial datasets, such as population density and land use patterns, to find the ideal greenspace site which would then be shown on a generated map.

Additionally, GeoBot could also be added to the Gazetteer for Scotland website as a geospatial information assistant. A limitation of this research is that GeoBot currently draws from a small PostgreSQL database, and so to make this a viable geospatial information assistant the full Gazetteer for Scotland Oracle database would ideally need to be vectorised and embedded.

Conclusion

This research confirmed that drawing upon GeoBot's RAG Model data, instead of original LLM training data, does improve LLM Scottish geographical responses, identifying local landmarks and Scottish locations missed by traditional LLM's. This is supported by GeoBot performing the strongest in response accuracy and similarity (Cosine Similarity and ROUGE Score), especially when compared to the base OpenAI GPT-4 and comparative no-code chatbot Botsonic.

The RAG model choice does also significantly affect the quality of geographical outputs, with GeoBot consistently outperforming Botsonic. While GeoBot did perform the strongest overall improvements would still be beneficial, especially in areas where GeoBot performed poorly, such as interface useability and response consistency. This also highlights that Botsonic is valuable for users who prioritise ease of use, suggesting that the two created RAG models are suitable for different use cases. Finally, GeoBot also evidenced that implementing map generation capabilities into a RAG Model is viable, although further improvements are needed to match geospatial industry standard.

Bibliography

Amazon Web Services (n.d.a). *What are Large Language Models? - LLM AI Explained - AWS*. Amazon Web Services, Inc. Available at: <https://aws.amazon.com/what-is/large-language-model/>. [Accessed: 3rd March 2025].

Amazon Web Services. (n.d.b). *Foundation Model API Service – Amazon Bedrock – AWS*. Amazon Web Services, Inc. Available at: <https://aws.amazon.com/bedrock/> [Accessed: 23rd July 2025].

Amazon Web Services (n.d.c). *What is Natural Language Processing? - NLP - AWS*. Amazon Web Services, Inc. Available at: <https://aws.amazon.com/what-is/nlp/> [Accessed: 23rd July 2025].

Amazon Web Services (n.d.d). *What is RAG? - Retrieval-Augmented Generation Explained - AWS*. Amazon Web Services. Available at: <https://aws.amazon.com/what-is/retrieval-augmented-generation/>. [Accessed: 3rd March 2025].

Amazon Web Services. (n.d.e). *What Are Transformers? - Transformers in Artificial Intelligence Explained*. Amazon Web Services, Inc. Available at: <https://aws.amazon.com/what-is/transformers-in-artificial-intelligence/> [Accessed: 23rd July 2025].

Amell, S. (2023). *How to Build a Private LLM: A Comprehensive Guide*. Medium. Available at: <https://medium.com/@iamamellstephen/how-to-build-a-private-llm-a-comprehensive-guide-296eae0e7db9> [Accessed: 23rd July 2025].

Amin, K.S., Mayes, L.C., Khosla, P., Doshi, R. (2024). Assessing the Efficacy of Large Language Models in Health Literacy: A Comprehensive Cross-Sectional Study. *The Yale journal of biology & medicine/Yale journal of biology and medicine*, 97(1), pp.17–27. DOI: <https://doi.org/10.59249/ztoz1966>.

Bashir, N., Donti, P., Cuff, J., Sroka, S., Ilic, M., Sze, V., Delimitrou, C., and Olivetti, E. (2024). *The Climate And Sustainability Implications of Generative AI*. MIT. Available at: https://mit-genai.pubpub.org/pub/8ulgrckc/release/2 open_in_new [Accessed: 22 May 2025].

Bayer, T. (2025). *Vector Search Demystified: A Guide to pgvector, IVFFlat, and HNSW*. DEV Community. Available at: <https://dev.to/cubesoftware/vector-search-demystified-a-guide-to-pgvector-ivfflat-and-hnsw-36hf> [Accessed: 27th May 2025].

Beautiful Soup. (n.d). *Beautiful Soup Documentation — Beautiful Soup 4.4.0 documentation*. Beautiful Soup. Available at: <https://beautiful-soup-4.readthedocs.io/en/latest/> [Accessed: 23rd July 2025].

Besen, S. (2024). *The Limitations and Advantages of Retrieval Augmented Generation (RAG)*. Towards Data Science. Available at: <https://towardsdatascience.com/the-limitations-and-advantages-of-retrieval-augmented-generation-rag-9ec9b4ae3729/> [Accessed: 3rd March 2025].

Botsonic. (n.d). *AI Chatbot for Website - Custom ChatGPT Chat Bots*. Botsonic. Available at: <https://botsonic.com>. [Accessed: 23rd July 2025].

Brown, S. (2021). *Machine Learning, Explained*. MIT Sloan. Available at: <https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained> [Accessed: 23rd July 2025].

Cambridge Dictionary. (2022). *Dataset*. Cambridge Dictionary. Available at: <https://dictionary.cambridge.org/dictionary/english/dataset> [Accessed 23rd July 2025].

Chawre, H. (2023). *Fine-Tuning LLMs: Overview, Methods, and Best Practices*. Turing. Available at: <https://www.turing.com/resources/finetuning-large-language-models> [Accessed: 20th June 2025].

Chen, Y., Wang, W., Lobry, S., Kurtz, C. (2024). An LLM Agent for Automatic Geospatial Data Analysis. ArXiv. Available at: <https://arxiv.org/abs/2410.18792> [Accessed: 3rd March 2025].

Chugani, V. (2024). *What is Cosine Distance?*. Datacamp. Available at: <https://www.datacamp.com/tutorial/cosine-distance> [Accessed: 23rd July 2025].

Cliffords, N., Holloway, S., Rice, S.P. and Valentine, G. (2008). *Key Concepts in Geography*. 2nd ed. SAGE Publications Ltd.

Cloudflare. (2022). *What is a large language model (LLM)?*. CloudFlare. Available at: <https://www.cloudflare.com/en-gb/learning/ai/what-is-large-language-model/> [Accessed: 23rd July 2025].

Cresswell, T. (2015) *Place: An Introduction*, 2nd Chichester: Wiley-Blackwell.

Crettaz, V. (2025). *A quick introduction to vector search*. Elasticsearch Labs. Available at: <https://www.elastic.co/search-labs/blog/introduction-to-vector-search> [Accessed 23 Jul. 2025].

Crossman-Smith, J. (2025). *Distance Metrics in Machine Learning - Low Code for Data Science*. Medium. Available at: <https://medium.com/low-code-for-advanced-data-science/distance-metrics-in-machine-learning-without-the-maths-994392d3995b> [Accessed: 23rd July 2025].

Devlin, J. and Chang, M. (2018). *Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing*. Available at: <https://research.google/blog/open-sourcing-bert-state-of-the-art-pre-training-for-natural-language-processing/> [Accessed: 3rd March 2025].

Escape Force. (2025). *Understanding Cosine Similarity and its Role in LLM Models with Retrieval-Augmented Generation (RAG)*. Escape Force. Available at: <https://www.escape-force.com/post/understanding-cosine-similarity-and-its-role-in-llm-models-with-retrieval-augmented-generation-rag>. [Accessed: 27th May 2025].

Esri. (n.d). *ArcGIS Field Maps | Field Apps & Mobile Data Collection*. Esri. Available at: <https://www.esri.com/en-us/arcgis/products/arcgis-field-maps/overview> [Accessed: 23rd July 2025].

Farquhar, S., Kossen, J., Kuhn, L. and Gal, Y. (2024). Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017), pp.625–630.
DOI:<https://doi.org/10.1038/s41586-024-07421-0>.

Fortinet. (2023). *What Is an API Key? | API Key Definition*. Fortinet. Available at: <https://www.fortinet.com/uk/resources/cyberglossary/api-key> [Accessed: 23rd July 2025].

FullStory. (2024). *What is Data Processing? Definition, Steps & Methods*. FullStory. Available at: <https://www.fullstory.com/blog/what-is-data-processing/> [Accessed: 23rd July 2025].

Gartner, D. (2025). *Why AI and Large Language Models Benefit from Cartography*. Esri. Available at: <https://www.esri.com/about/newsroom/arcnews/why-ai-and-large-language-models-benefit-from-cartography> [Accessed 22nd May 2025].

Geographical Association. (n.d). *Concepts in geography*. Geographical Association. Available at: <https://geography.org.uk/ite/initial-teacher-education/geography-support-for-trainees-and-ects/learning-to-teach-secondary-geography/geography-subject-teaching-and-curriculum/geography-knowledge-concepts-and-skills/concepts/concepts-in-geography/>. [Accessed: 3rd March 2025].

Ghosh, B. (2024). *When to Apply RAG vs Fine-Tuning*. Medium. Available at: <https://medium.com/@bijit211987/when-to-apply-rag-vs-fine-tuning-90a34e7d6d25> [Accessed: 23rd July 2025].

Gittings, B. (n.d). *Gazetteer for Scotland*. The Gazetteer For Scotland. Available at: <https://www.scottish-places.info>. [Accessed: 4th of March 2025].

Google Cloud Platform. (n.d). *Cloud Computing Services*. Google Cloud. Available at: <https://cloud.google.com/?hl=en> [Accessed: 23rd July 2025].

Google Maps (n.d). *Google Maps*. Google Maps. Available at: <https://www.google.com/maps> [Accessed: 23rd July 2025].

Google. (2023). *Gemini*. Google. Available at: <https://gemini.google.com/app>. [Accessed: 3rd March 2025].

Google. (2025). *Vertex AI RAG Engine overview*. Google. Available at: <https://cloud.google.com/vertex-ai/generative-ai/docs/rag-engine/rag-overview> [Accessed: 26th May 2025].

Google. (n.d). *What Is Semantic Search?*. Google. Available at: <https://cloud.google.com/discover/what-is-semantic-search> open_in_new. [Accessed: 22nd May 2025].

GPTHub. (2022). *OpenAI Unveils a Powerful, Cost-Effective, and User-Friendly Embedding Model*. GPTHub. Available at: <https://www.gpthub.ai/ai-news-and-announcements/entrepreneurship-ideas-for-young-and-ambitious-people/> [Accessed 9th June 2025].

Grigoryan, S. (n.d). *User Feedback Survey: Types, Best Practices & 20 Questions to Ask*. Userpilot Blog. Available at: <https://userpilot.com/blog/user-feedback-survey-saas/> [Accessed: 23rd July 2025].

Harvard University. (2025). *Generative Artificial Intelligence (AI)*. Harvard University. Available at: <https://www.huit.harvard.edu/ai> [Accessed: 23rd July 2025].

Hernández, J. (2023). *What are Vector Embeddings in AI and LLM? - Jaime Hernández - Medium*. Medium. Available at: <https://devjaime.medium.com/what-are-vector-embeddings-in-ai-and-llm-d40732c27f82> [Accessed: 23rd July 2025].

Herroros, Q. Veasey, T., Papaoikonomou, T. (2023). *RAG evaluation metrics: A journey through metrics*. Search Labs. Available at: <https://www.elastic.co/search-labs/blog/evaluating-rag-metrics> [Accessed: 23rd July 2025].

HLEG. (n.d). *Expert group on AI | Shaping Europe's Digital Future*. High-Level Expert Group on Artificial Intelligence. Available at: <https://digital-strategy.ec.europa.eu/en/policies/expert-group-ai> [Accessed: 23rd July 2025].

Hopsworks. (2023). *What is Similarity Search?*. Hopsworks. Available at: <https://www.hopsworks.ai/dictionary/similarity-search> [Accessed: 27th May 2025].

HPE. (2024). *What is Data Storage? | Glossary*. HPE. Available at: <https://www.hpe.com/uk/en/what-is/data-storage.html> [Accessed: 23rd July 2025].

IBM (2021). *Geospatial data*. IBM. Available at: <https://www.ibm.com/think/topics/geospatial-data> [Accessed: 23rd July 2025].

International Geosphere-Biosphere Programme. (n.d). *Earth system definitions*. IGBP. Available at: <http://www.igbp.net/globalchange/earthsystemdefinitions.4.d8b4c3c12bf3be638a80001040.html>. [Accessed: 3rd March 2025].

Jadeja, M. (2022). *Jaccard Similarity Made Simple: A Beginner's Guide to Data Comparison*. Medium. Available at: <https://mayurdhvajsinhjadeja.medium.com/jaccard-similarity-34e2c15fb524> [Accessed: 23rd July 2025].

Kapuściński, M. (2024). *The Evolution of ChatGPT from OpenAI: From GPT-1 to GPT-4o | TTMS*. Transition Technologies MS S.A. Available at: <https://ttms.com/uk/chat-gpt-evolution/> [Accessed: 14th April 2025].

Karabiber, F. (n.d). *Cosine Similarity*. Learn Data Science. Available at: <https://www.learndatasci.com/glossary/cosine-similarity/> [Accessed: 23rd July 2025].

Komprise. (2019). *Data Retrieval*. Komprise. Available at: https://www.komprise.com/glossary_terms/data-retrieval/ [Accessed: 23rd July 2025].

Lange, K., Onyegbula, B. (2025). *The Role of Prompt Engineering in Useful AI: Benefits, Techniques, and Applications for Better Prompting*. Splunk. Available at: https://www.splunk.com/en_us/blog/learn/prompt-engineering.html [Accessed: 24th July 2025].

Leaflet. (2019). *Leaflet — an open-source JavaScript library for interactive maps*. Leaflet. Available at: <https://leafletjs.com> [Accessed: 23rd July 2025].

Lenovo. (n.d). *Operating System: What is an Operating System and How Does it Work?*. Lenovo UK. Available at: <https://www.lenovo.com/gb/en/glossary/operating-system/?orgRef=https%253A%252F%252Fwww.google.com%252F> [Accessed: 23rd July 2025].

Li, P., Yang, J., Islam, M.A., Ren, S. (2023). *Making AI less “thirsty”: Uncovering and addressing the secret water footprint of AI models*. ArXiv. Available at: <https://arxiv.org/pdf/2304.03271> open_in_new [Accessed: 23rd July 2025].

Manvi, R., Khanna, S., Burke, M., Lobell, D., Ermon, S. (2024). *Understanding Biases in Large Language Models Through Geography*. ArXiv. Available at: <https://doi.org/10.48550/arXiv.2402.02680>. [Accessed: 22nd May 2025].

Matos, T. (2024). *Evaluating Retrieval-Augmented Generation (RAG) Models: A Comprehensive Guide*. Medium. Available at: <https://medium.com/@rtales/evaluating-retrieval-augmented-generation-rag-models-a-comprehensive-guide-5205c6f1eefa> [Accessed: 3rd March 2025].

McGrath, A. and Jonker, A. (2024). *LLM evaluation*. IBM. Available at: <https://www.ibm.com/think/insights/llm-evaluation> [Accessed: 7th June 2025].

McKinsey. (2024). *What is Artificial General Intelligence (AGI)?*. McKinsey and Company. Available at: <https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-artificial-general-intelligence-agi>. [Accessed: 23rd July 2025].

Mckirdy, A., Crofts, R. (2010). *Scotland: The Creation of its Natural Landscape A Landscape Fashioned by Geology*. Scottish Geology Trust. Available at: <https://www.scottishgeologytrust.org/downloads/LandscapeFashionedByGeology-scotland.pdf> [Accessed: 25th July 2025].

Microsoft. (n.d.a). *Get your document's readability and level statistics - Microsoft Support*. Microsoft. Available at: <https://support.microsoft.com/en-gb/office/get-your-document-s-readability-and-level-statistics-85b4969e-e80a-4777-8dd3-f7fc3c8b3fd2> [Accessed: 23rd July 2025].

Microsoft. (n.d.b). *Microsoft Forms | Surveys, Polls, and Quizzes*. Microsoft. Available at: <https://www.microsoft.com/en-gb/microsoft-365/online-surveys-polls-quizzes> [Accessed: 23rd July 2025].

Microsoft. (2024). *Azure OpenAI Service - Azure OpenAI*. Microsoft. Available at: <https://learn.microsoft.com/en-us/azure/ai-services/openai/concepts/prompt-engineering?tabs=chat> [Accessed: 23rd July 2025].

Mohandas, G. and Moritz, P. (2023). *Building RAG-based LLM Applications For Production*. Anyscale. Available at: https://www.anyscale.com/blog/a-comprehensive-guide-for-building-rag-based-llm-applications-part-1-open_in_new [Accessed: 22nd May 2025].

Morozov, E. (2023). The Problem with Artificial Intelligence? It's Neither Artificial nor Intelligent. *The Guardian*. Available at: <https://www.theguardian.com/commentisfree/2023/mar/30/artificial-intelligence-chatgpt-human-mind> [Accessed: 23rd July 2025].

Mucci, T., Stryker, C. (2023). *Artificial Superintelligence*. IBM. Available at: <https://www.ibm.com/think/topics/artificial-superintelligence> [Accessed: 23rd July 2025].

Nagella, V. (2019). *Cosine Similarity Vs Euclidean Distance*. Medium. Available at: <https://medium.com/@sasi24/cosine-similarity-vs-euclidean-distance-e5d9a9375fc8> [Accessed: 23rd July 2025].

Nancholas, B. (2023). *Narrow artificial intelligence: advantages, disadvantages, and the future of AI*. University of Wolverhampton. Available at: <https://online.wlv.ac.uk/narrow-artificial-intelligence-advantages-disadvantages-and-the-future-of-ai/> [Accessed: 23rd July 2025].

National Geographic. (n.d). *Place*. National Geographic. Available at: <https://education.nationalgeographic.org/resource/place/#>. [Accessed: 3rd March 2025].

National Library of Medicine. (2022). *Web Scraping*. National Library of Medicine. Available at: <https://www.nlm.gov/guides/data-glossary/web-scraping> [Accessed: 23rd July 2025].

Neural Technologies. (2024). *Unleashing the Power of Self-Learning AI Models with ActivML*. Neuralt. Available at: <https://www.neuralt.com/news-insights/unleashing-the-power-of-self-learning-ai-models-with-activml> [Accessed 23rd July 2025].

Ni, K. (2024). *Exploring LLM Stability: Insights from a Detailed Analysis*. Medium. Available at: <https://medium.com/@kai.ni/exploring-llm-stability-insights-from-a-detailed-analysis-950365924176> [Accessed: 7th June 2025].

Office for National Statistics. (2022). *Percentage of homes and individuals with technological equipment*. Office For National Statistics. Available at: <https://www.ons.gov.uk/aboutus/transparencyandgovernance/freedomofinformationfoi/percentageofhomesandindividualswithtechnologicalequipment> [Accessed: 23rd July 2025].

OpenAI. (2021). *GPT-4*. OpenAI. Available at: <https://openai.com/index/gpt-4-research/> [Accessed: 23rd July 2025].

OpenAI. (2022). *New and improved embedding model*. OpenAI. Available at: <https://openai.com/index/new-and-improved-embedding-model/>.

OpenAI. (2024). *OpenAI Platform*. OpenAI. Available at: <https://platform.openai.com/docs/guides/prompt-engineering>. [Accessed: 3rd March 2025].

OpenAI. (2025). *ChatGPT*. OpenAI. Available at: <https://openai.com/chatgpt/overview/> [Accessed: 23rd July 2025].

Oracle. (2014). *What is a relational database?* Oracle. Available at: <https://www.oracle.com/uk/database/what-is-a-relational-database/> [Accessed: 23rd July 2025].

Ozkaya, M. (2024). *Understanding Semantic Meaning and Similarity Search: Cosine similarity and Euclidean distance*. Medium. Available at: <https://mehmetozkaya.medium.com/understanding-semantic-meaning-and-similarity-search-cosine-similarity-and-euclidean-distance-666fcb5911ea> open_in_new [Accessed: 22nd May 2025].

Paris, M. (2025). *ChatGPT Hits 1 Billion Users? ‘Doubled In Just Weeks’ Says OpenAI CEO*. Forbes. Available at: <https://www.forbes.com/sites/martineparis/2025/04/12/chatgpt-hits-1-billion-users-openai-ceo-says-doubled-in-weeks/> [Accessed: 23rd July 2025].

Pinecone. (n.d). *Vector Similarity Explained*. Pinecone. Available at: <https://www.pinecone.io/learn/vector-similarity/> [Accessed: 23rd July 2025].

Poulton, N. (2024). *AI Explained : facts, fiction & future*. Birmingham: Packt Publishing, Limited.

Quinn, A., Lecker, G. (2024). *LLM Metrics: Key Metrics Explained*. Iguazio. Available at: <https://www.iguazio.com/blog/llm-metrics-key-metrics-explained/> [Accessed: 7th June 2025].

RAGAS. (n.d). *Semantic Similarity*. RAGAS. Available at: https://docs.ragas.io/en/latest/concepts/metrics/available_metrics/semantic_similarity/ [Accessed 7th June 2025].

Readable. (2024). *Flesch Reading Ease and the Flesch Kincaid Grade Level*. Readable. Available at: <https://readable.com/readability/flesch-reading-ease-flesch-kincaid-grade-level/> [Accessed: 27th July 2025].

Readable. (n.d). *The Automated Readability Index*. Readable. Available at: <https://readable.com/readability/automated-readability-index/> [Accessed: 23rd July 2025].

Ribas, M. (2024). LLM Crystal Ball: Future of LLM Development. Turing. Available at: <https://www.turing.com/blog/llm-crystal-ball-future-of-llm-development#> [Accessed: 3rd March 2025].

Ritchie, H. (2025). *What's the carbon footprint of using ChatGPT?*. Sustainability by Numbers. Available at: <https://www.sustainabilitybynumbers.com/p/carbon-footprint-chatgpt> [Accessed: 23rd July 2025].

Roberts, J., Lüddecke, T., Das, S., Han, K., Albanie, S. (2023). *GPT4GEO: How a Language Model Sees the World's Geography*. ArXiv. Available at: <https://arxiv.org/abs/2306.00020> [Accessed: 3rd March 2025].

SAP. (2024). *What is prompt engineering?*. SAP. Available at: <https://www.sap.com/resources/what-is-prompt-engineering> [Accessed: 24th July 2025].

SAS. (2010). *Artificial Intelligence – What it is and why it matters*. SAS. Available at: https://www.sas.com/en_gb/insights/analytics/what-is-artificial-intelligence.html [Accessed: 25th July 2025].

Sandu, C. (2024). *Attention Mechanism*. Medium. Available at: <https://medium.com/@calin.sandu/attention-mechanism-294a63541d95> open_in_new [Accessed: 22nd May 2025].

Santhosh, S. (2023). *Understanding BLEU and ROUGE score for NLP evaluation*. Medium. Available at: <https://medium.com/@sthanikamsanthosh1994/understanding-bleu-and-ROUGE-score-for-nlp-evaluation-1ab334ecadcb> [Accessed: 23rd July 2025].

Sarker, I.H. (2021). Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*, 2(3), pp.1–21. DOI:<https://doi.org/10.1007/s42979-021-00592-x>.

Scheider, S., Richter, K.-F. (2023). Pragmatic GeoAI: Geographic Information as Externalized Practice. *KI - Künstliche Intelligenz*, 37(1), pp.17–31. DOI: <https://doi.org/10.1007/s13218-022-00794-2>.

Schwaber-Cohen, R. (2023). *Chunking Strategies for LLM Applications*. Pinecone. Available at: <https://www.pinecone.io/learn/chunking-strategies/>. [Accessed: 25th June 2025].

Scott, B. (2023). *How to Use the Automated Readability Index (ARI) for Clearer Communication*. ReadabilityFormulas. Available at: <https://readabilityformulas.com/the-automated-readability-index-ari/> [Accessed: 23rd July 2025].

Selenium. (n.d). *SeleniumHQ Browser Automation*. Selenium. Available at: <https://www.selenium.dev> [Accessed: 23rd July 2025].

Sheikh, H., Prins, C., Schrijvers, E. (2023). *Mission AI: The New System Technology*. 1st ed. 2023. Cham: Springer Nature.

Shone, O. (2025). *5 key features and benefits of retrieval augmented generation (RAG)*. The Microsoft Cloud Blog. Available at: <https://www.microsoft.com/en-us/microsoft-cloud/blog/2025/02/13/5-key-features-and-benefits-of-retrieval-augmented-generation-rag/> [Accessed: 9th June 2025].

Siles. (2024). *Exploring the evolution of ChatGPT: From GPT-3 to GPT-4*. Telefónica. Available at: <https://www.telefonica.com/en/communication-room/blog/exploring-evolution-chatgpt-gpt-3-gpt-4/> [Accessed: 14th April 2025].

Tatarek, M. (2023). *Costs and benefits of your own LLM*. Medium. Available at: <https://medium.com/@maciej.tatarek93/costs-and-benefits-of-your-own-llm-79f58c0eb47f>. [Accessed: 23rd July 2025].

The University of Edinburgh. (2024). *Introduction to ELM*. ELM. Available at: <https://information-services.ed.ac.uk/computing/comms-and-collab/elm/elm-competence-centre/introduction-to-elm> [Accessed: 23rd July 2025].

Tsementzis, D. (2024). *Will AI lead to superintelligence or just super-automation?*. Goldman Sachs. Available at: <https://www.goldmansachs.com/insights/articles/will-ai-lead-to-superintelligence-or-just-super-automation> [Accessed: 23rd July 2025].

Verma, P., Tan, S. (2024). *A bottle of water per email: the hidden environmental costs of using AI chatbots*. Washington Post. Available at: <https://www.washingtonpost.com/technology/2024/09/18/energy-ai-use-electricity-water-data-centers/> [Accessed: 23rd July 2025].

Woyera. (2024). *We tested the top 7 no-code chat bot tools. Here's what we found.* Medium. Available at: <https://medium.com/@woyera/guide-to-no-code-ai-chatbots-40d37bc6d56b> [Accessed: 27th May 2025].

Yashwanth, S. (2025). *Retrieval Augmented Generation (RAG) 05 :- Indexing in LLM RAG.* Medium. Available at: https://medium.com/@yashwanths_29644/retrieval-augmented-generation-rag-05-indexing-strategies-4b822513ea07 [Accessed 25th June 2025].

Yu, D., Bao, R., Mai, G., Zhao, L. (2025). *Spatial-RAG: Spatial Retrieval Augmented Generation For Real-World Spatial Reasoning Questions.* ArXiv. Available at: <https://arxiv.org/abs/2502.18470v2> open_in_new [Accessed: 22nd May 2025].

Zhang, Y., Wei, C., He, Z., Yu, W. (2024). GeoGPT: An assistant for understanding and processing geospatial tasks. *International Journal of Applied Earth Observation and Geoinformation*, 131, pp.103976–103976. DOI :<https://doi.org/10.1016/j.jag.2024.103976>.

Zhou, C. (2024). *Understanding Jaccard Similarity: A Powerful Tool for Data Analysis.* Medium. Available at: <https://medium.com/@conniezhou678/understanding-jaccard-similarity-a-powerful-tool-for-data-analysis-42abaaafd782> [Accessed: 23rd July 2025].

Part II: Technical Report

The Logical Architecture of RAG Model GeoBot

Table of Contents

List of Figures.....	2
List of Tables	4
Chapter 1: Introduction	5
Chapter 2: RAG Model System Justification	7
Alternative LLM Optimisation Methodologies.....	10
Alternatives to a Python coded RAG Model	15
Alternative LLM Models	18
Chapter 3: GeoBot's Logical Architecture Overview	19
Chapter 4: GeoBot's Backend Architecture.....	21
GeoBot's Database Methodology	21
Data Selection	21
Data Storage.....	26
Data Processing.....	31
Flask App Methodology	40
Processing: Similarity Searching	41
Results: GeoBot Response Generation.....	42
Map Methodology.....	45
Chapter 5: GeoBot's Frontend Architecture	49
Frontend Architecture Overview.....	49
Interface Design Methodology and Justification	50
Interface Limitations and Improvements	50
Alternative Interface Methodology	51
Chapter 6: GeoBot Security Considerations.....	52
Chapter 7: Results Methodology	54
Response Relevance, Accuracy and Clarity Evaluation Overview (Appendix F)	55
Response Relevance (Semantic Similarity).....	55
Response Accuracy	57
Response Clarity.....	58
Task Success/Fail Rate and Hallucination Rate.....	60
Overall Result Strengths, Limitations and Improvements	61
Chapter 8: Technical Report Conclusion.....	62
Bibliography.....	63
Appendices	75

List of Figures

Figure 1: Geobot's Logical Architecture, P6

Figure 2: GPT-4 And GPT-3. Comparative Test Results (Openai, 2023), P9

Figure 3: Transformer Architecture (Gal, 2023), P13

Figure 4: GCP RAG Model Chatbot Workflow (Didonato, 2024), P15

Figure 5: Amazon Web Services Bedrock RAG Workflow (Saeed, 2024), P15

Figure 6: Geobot Full Process Workflow, P20

Figure 7: Geobot File Structure, P21

Figure 8: Haymarket Scotland For Gazetteer Entry (Gittings, 2023), P22

Figure 9: Selenium Web Scraping Workflow, P25

Figure 10: Web Scraping Original And Suggested Improved Web Scraping Methodology, P25

Figure 11: Postgresql Table Creation Commands, P28

Figure 12: E-R Diagram For Proposed Postgresql Tables. PK (Primary Key) And FK (Foreign Key), P29

Figure 13: Proposed Postgresql Table SQL Creation Code, P29

Figure 14: Elasticsearch RAG Workflow (Elasticsearch, N.D), P31

Figure 15: Geobot SQL Chunking Code, P33

Figure 16: Excerpt From The Gazetteer For Scotland Entry 'The Meadows' (Gittings, 2025), P34

Figure 17: Embedding Process (Ashutosh, 2024), P35

Figure 18: Embedding Python Script Methodology, P36

Figure 19: IVFFLAT Indexing Structure (Emanuilov, 2024), P38

Figure 20: Indexing SQL Geobot Code, P38

Figure 21: HNSW Indexing Structure (Pinecone, N.D), P39

Figure 22: Geobot Flask App Workflow, P41

Figure 23: Geobot Euclidean Distance Code Excerpt (Appendix C), P42

Figure 24: Role Prompting Criteria (Information From Cheung, 2024), P43

Figure 25: Geobot Role Prompting Code (Appendix C), P44

Figure 26: Geobot Conversation History Showcase, P45

Figure 27: Overall Map Creation Methodology (Code In Appendix C), P46

Figure 28: Measurement Tool (Left), Transport Options Tile Layer (Centre) And Pop-Up Markers (Right), P47

Figure 29: Route Calculations By Car (Left), Walking (Centre) And Cycling (Right), P47

Figure 30: Mapbox 3d Model (Left) And Digital Mapping (Right) Showcase, P48

Figure 31: Alternative Methodology Cartopy Map Code Extract, P49

Figure 32: Geobot Main Website Interface, P51

Figure 33: Geobot Login Page Interface, P51

Figure 34: Chatbot UI Design (Information From IBM, 2025), P52

Figure 35: Cosine Similarity Equation (Varun, 2020), P57

Figure 36: Cosine Similarity Geobot Test Code (Appendix F), P58

Figure 37: ROUGE Score Test Code (Appendix F), P59

Figure 38: Flesch Reading Ease Test Code, P60

Figure 39: Microsoft Forms User Feedback Survey Relevant Questions, P62

Figure 40: Microsoft Forms User Feedback Survey Map Generation Questions, P63

List of Tables

Table 1: RAG Model Types Descriptions (Jikadara, 2025), P7

Table 2: GPT-3 And GPT-4 Comparison (Botpress, 2025), P8

Table 3: Summary Of Alternative LLM Optimisation Techniques, P10

Table 4: Prompt Engineering Concepts (Aryani, 2023), P11

Table 5: Supervised Fine-Tuning Techniques (Data From Chawre, 2023), P12

Table 6: Reinforcement Learning Fine Tuning Techniques (Data From Chawre, 2023), P12

Table 7: Pairwise Comparison. 1 = First Model Preferred, 0.5 = Equal, 0 = Second Model Preferred, P16

Table 8: No Code Platform Chatbot Rankings (Woyera, 2024), P17

Table 9: LLM Comparison (Information From Meta (2025)), P18

Table 10: Alternative Data Selection Methodologies, P23

Table 11: Main Web Scraping Tools (Heath, 2023), P26

Table 12: Geobot Postgresql Initial Table Structure, P27

Table 13: Data Storage Alternative Methodologies, P30

Table 14: Chunking Methods In Order Of Complexity (Mishra (2024) And Schwaber-Cohen (2023)), P32

Table 15: Example Chunking Output (Description Summarised), P34

Table 16: Openai Embedding Model Comparison (Kutz, 2025), P37

Table 17: Geobot Postgresql Table Structure After Chunking, Embedding And Indexing, P40

Table 18: Security Consideration Methodology And Justification, P54

Table 19: RAG Evaluation Metrics (Zhang, 2024) And Justification For Inclusion, 56

Chapter 1: Introduction

This research demonstrated that Retrieval Augmented Generation (RAG) Model GeoBot, and to some extent the comparative RAG Model Botsonic, enhanced Large Language Model (LLM) Scottish geographical responses, as well as successfully integrating map generation features. A RAG Model improves LLM outputs by referencing a provided dataset instead of relying on LLM training data (Amazon Web Services, n.d). LLMs are models trained on text datasets to generate responses (CloudFlare, 2022).

GeoBot utilised OpenAI's latest LLM GPT-4 (Generative Pre-trained Transformer Model) to generate responses. It drew upon data from the Gazetteer for Scotland, a comprehensive database of Scottish locations (Gittings, n.d), also integrating a map feature. As a comparison, a customizable, no-code chatbot was created using the Botsonic (n.d) platform that also integrates data from the Gazetteer for Scotland to create responses through a Retrieval Augmented Generation (RAG) approach.

The Technical Report will demonstrate the logical architecture (technical layout) of GeoBot. Figure 1 shows the logical architecture of GeoBot, the main RAG model created for this research. This architecture can be broadly summarised into the Frontend user interface (GeoBot interactive website and map generation) and the Backend (data storage, processing and retrieval, followed by LLM response generation). The aim of GeoBot was to produce a cost-effective, scalable and repeatable RAG Model. Each chapter provides an overview of a component of GeoBot, evaluating the technological methodology used, including it's strengths, weaknesses, limitations, alternative options and potential improvements.

This report goes beyond the research paper to analyse GeoBot's methodology in detail. Chapter Two focuses on the RAG model system, providing a methodology overview and the alternatives that were considered. Chapter Three, Four and Five then analyse the Backend and Frontend logical architecture of GeoBot, with the methods employed to secure GeoBot against online attacks also analysed in Chapter Six. Finally, the results methodology, an essential section of the research which successfully evaluated GeoBot's responses, is analysed and critiqued in Chapter Seven.

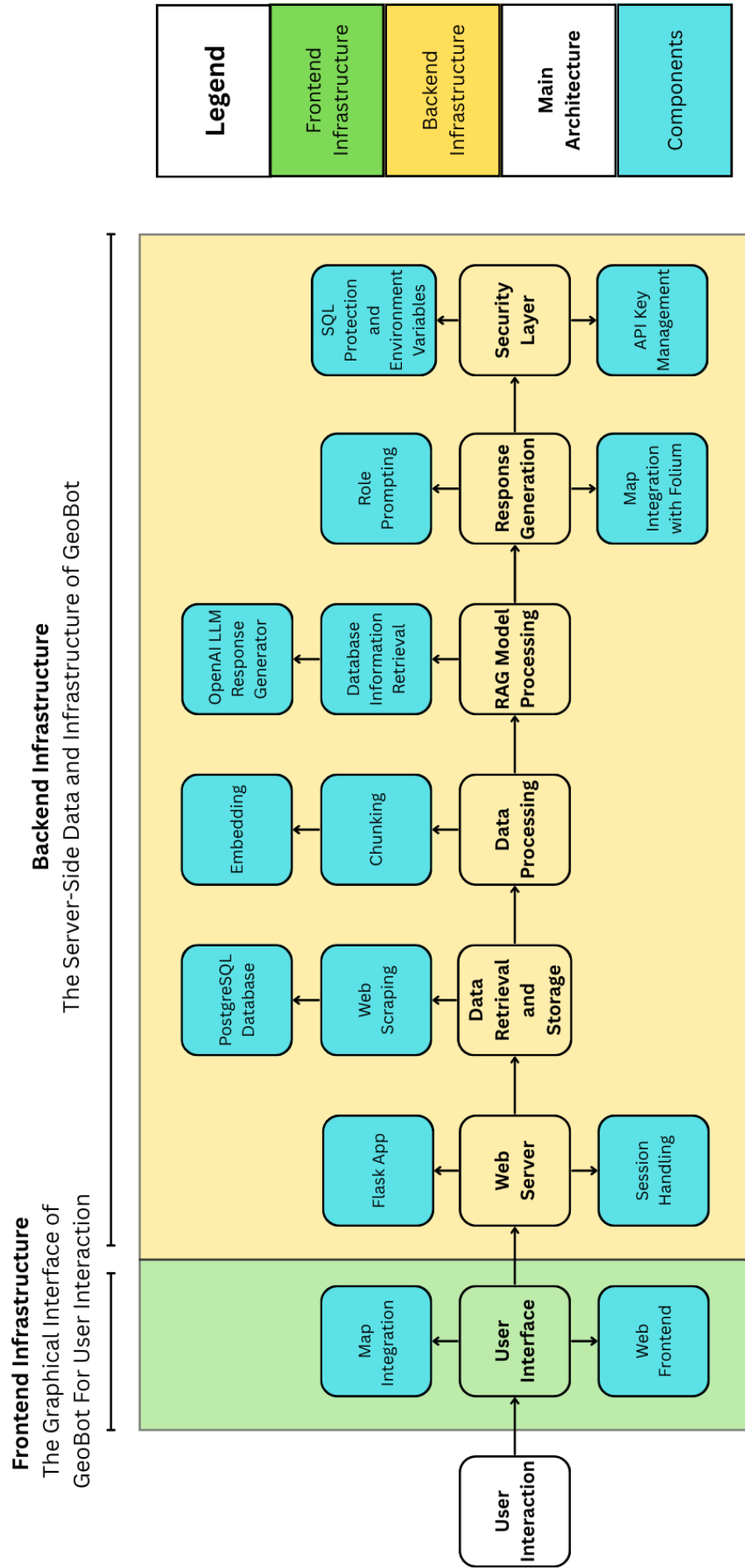


Figure 1: GeoBot's Logical Architecture

Chapter 2: RAG Model System Justification

This chapter provides a justification of the RAG Model methodology, then examining alternatives to this approach for improving the Scottish geographical responses of GeoBot. Although a RAG Model methodology was ultimately selected, other LLM optimisation techniques such as Direct LLM Information Input, Prompt Engineering, Fine-Tuning, and creating a Custom LLM Model are considered. The alternatives to coding a Python programming language-based RAG model specifically, such as Cloud Platforms and No-Code RAG chatbots are also discussed. Finally, the decision to incorporate OpenAI, instead of another LLM Model, is justified and alternatives are explored.

RAG Model Justification

A RAG Model approach was utilised to achieve a dramatic improvement in the accuracy and relevance of OpenAI's Scottish geographical responses by drawing upon updated external data, which reduces hallucinations (where LLM's produce responses that are incorrect), and ensures a cost-effective and secure external database.

GeoBot draws upon an external database that can be updated (Shone, 2025), meaning that it does not purely depend on the original LLM training data (the base information that LLM's draw upon) for its responses. This substantially reduces costs as it does not need multiple training stages to incorporate this new information directly into GeoBot (MacDonald, 2024), instead just drawing from a single external database. This external database is also not added to OpenAI's training data itself, meaning sensitive data is not exposed to OpenAI (Besen, 2024).

Table 1 shows the three main types of RAG models: Standard/Naive RAG, Advanced RAG and Modular RAG. The Standard RAG methodology was chosen for this research as speed, scalability and user experience are prioritised, whereas the other models focus on high customisation, improved retrieval and generation but with far less scalability and useability (Harkar, 2025).

Table 1: RAG Model Types Descriptions (Jikadara, 2025)

RAG Model	Description
Naïve/Standard RAG	A straightforward system involving indexing, retrieval and response generation using common similarity measures. Strongest for chatbot methodologies.
Advanced RAG	Enhanced standard model with techniques such as document re-ranking and compressing the prompt.
Modular RAG	Splits processes into different modules to combine different approaches for high customization.

A Python coded RAG model, instead of using a No-Code platform, was specifically chosen due to the high customisation capabilities that this approach enables, despite it being more methodologically complex. Python is an interpreted, object-oriented, high-level programming language (Python, n.d), and was chosen due to its ease of integration; it can work with a variety of external databases and LLM API Keys (a code to identify and authenticate a user (Fortinet, n.d)). Python made it seamless to program GeoBot to draw upon an external database and use OpenAI's API Key.

After choosing the Python coded RAG Model approach, the RAG Model LLM that constructs the GeoBot responses also had to be considered. Including an LLM in the RAG Methodology is vital so that GeoBot can produce logical and informed responses. The LLM component generates responses that are informed by both the retrieved RAG data and the broader context. Without including an LLM, GeoBot could technically produce a response, but it would not be intelligent and would purely be word-for-word extracts from the external database. The main RAG Model created for this research, GeoBot, utilises OpenAI's GPT-4 LLM, yet RAG Models can be implemented into any LLM agent (Google, 2025).

OpenAI is an AI research and deployment company (OpenAI, 2024) who have released a variety of LLM models. GPT-4 is a large closed-source generative pre-trained transformer model, which can intake generate both text and images (multimodal) (GPTHub, 2022). OpenAI's GPT-4 was purely chosen due to data security. In this research OpenAI was accessed through ELM, an Edinburgh Access to Large Language Model. This means that any queries entered into GeoBot are not shared with OpenAI as it operates within the University of Edinburgh's secure infrastructure. Without ELM, the information entered into GPT-4 is used for further model training (Cook, 2024), which is not preferable when handling copyrighted information such as for the Gazetteer for Scotland.

Compared to the previous version GPT-3, GPT-4 also offers increased size, capability and memory (Table 2). Figure 2 evidences the reasoning improvement of GPT-4, with the LLM scoring significantly higher in simulated exams compared to its predecessor GPT-3. While this was not a critical factor as GPT-4 was already chosen due to the ELM system, it means that GeoBot's responses were more accurate than if GPT-3 was utilised.

Table 2: GPT-3 and GPT-4 comparison (BotPress, 2025)

Version	Size	Modality	Conversation Memory	Price
GPT-3	175 billion parameters	Text	2048 Tokens	Free
GPT-4	1.8 trillion parameters	Text and Images	128k Tokens	Free

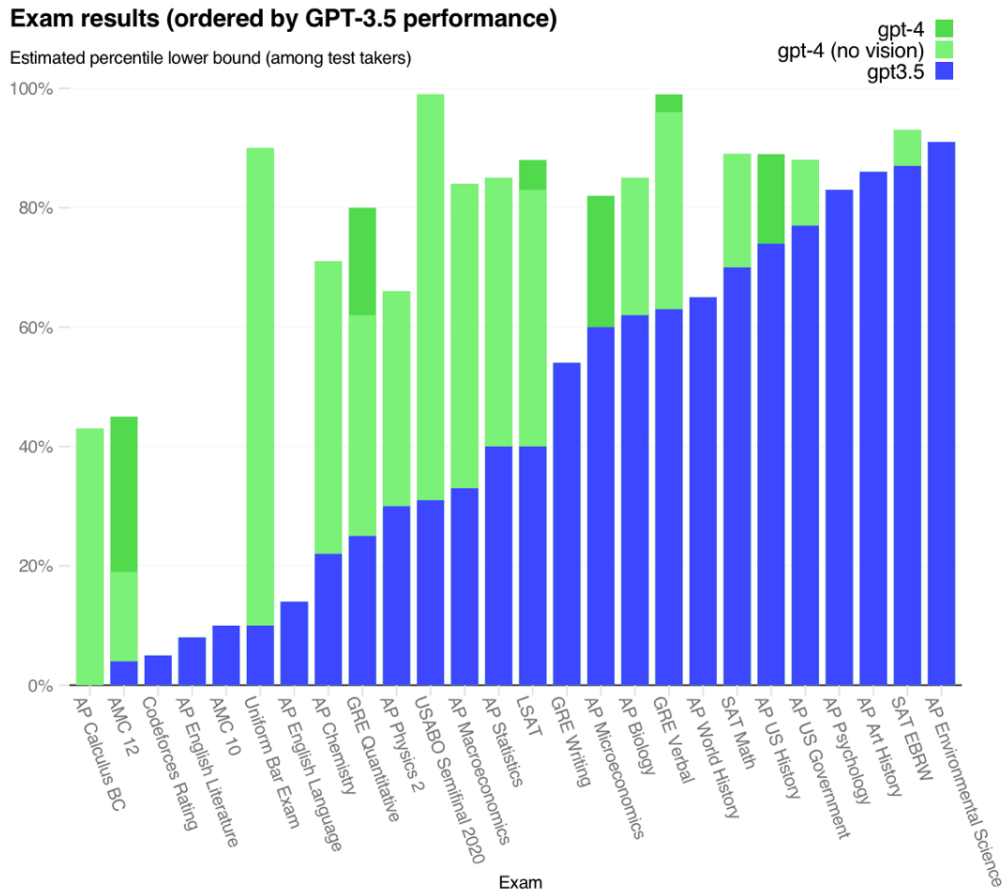


Figure 2: GPT-4 and GPT-3 Comparative Test Results (OpenAI, 2023)

Nevertheless, there are still weaknesses of the RAG methodology that must be considered. Barnett *et al* (2024) state that RAG Models often face challenges when accurately answering queries. They can product incorrect responses when the database is not explicitly clear, such as if the database contains vague information or when the top-ranked documents omit key information. They also can return responses in the wrong format, such as a list instead of a table and incorrectly answer queries due to contradicting database information. However, these issues can be substantially mitigated by improving the external database quality and the data retrieval methodology, both options which were implemented into GeoBot to ensure high quality responses.

Alternative LLM Optimisation Methodologies

Despite ultimately selecting a RAG Model methodology for this research, there are various techniques available for improving Scottish geographical LLM responses. The alternatives explored are listed in Table 3, and vary significantly in cost, methodology complexity and rarity. While these alternatives were ultimately not selected for GeoBot, they are still important to analyse here to both further justify the RAG Model implementation for GeoBot and to recognise their strengths as alternative approaches.

Table 3: Summary of Alternative LLM Optimisation Techniques

Technique	Cost	Methodology Complexity	Rarity
Direct LLM Input	Least Expensive	Least Complex	Common Use
Prompt Engineering	Small Cost	Least Complex	Common Use
RAG Model	Small Cost	Less Complex	Common Use
Fine Tuning	Moderate Cost	Moderately Complex	Occasional Use
Custom LLM Model	Most Expensive	Most Complex	Very Rare Use

Direct LLM Input

The easiest and cost-effective option is simply to directly input all the required data into an LLM such as OpenAI's ChatGPT. This input is stored in a context window (the LLM's working memory) that the LLM draws upon. This was considered as this approach can lead to fewer hallucinations and more informed responses (Bergmann, 2024). However, risks such as sensitive data disclosure, unauthorised access by malicious actors and a lack of data use transparency drastically decrease the feasibility of this approach. For example, the Meta AI Chatbot's (Meta, 2025) unclear data privacy settings have led to voice messages and apparently private prompt history being made available online (Gepiti, 2025). DeepSeek, a Chinese LLM, also stores personal user data such as email address, IP address and keystroke patterns (Gerken, 2025), potentially causing concern if the data was sensitive. This data security concern made this approach inviable for GeoBot.

Prompt Engineering

Prompt Engineering is the process of designing LLM user inputs to produce better outputs (McKinsey, 2024). This was considered as specifying the steps required for a successful answer to an LLM, such as providing examples and creating subtasks, can improve LLM outputs and subsequently Scottish geographical responses (McKinsey, 2024). It also requires no coding (not technically complex) and is very cost effective: the cost of a single query to an LLM is very minimal and often free (OpenAI's GPT-4 model costs approximately between \$0.01 and \$0.1 per user query (Iterations, 2025)). Table 4 shows the variety of Prompt Engineering techniques available.

Table 4: Prompt Engineering Concepts (Aryani, 2023)

Prompt Engineering Concept	Explanation
Zero-Shot Learning	Stating the prompt without assuming any prior knowledge.
One-Shot Learning	Providing one example with the prompt.
Few-Shot Learning	Providing a few examples with the prompt.
Chain-of-Thought Prompting	The prompt includes a step-by-step thought process.
Iterative Prompting	Refining the prompts based on the outputs.
Negative Prompting	Specifying what the response must not include in your prompt.
Hybrid Prompting	Combining different prompt methods.
Prompt Chaining	Giving multiple prompts to break the problem down into smaller subsections.

However, ultimately this approach was not chosen for GeoBot because it cannot address deeper inaccuracies or gaps in the LLM original training data, unlike a RAG Model which can utilise external databases. It mainly alters output style rather than significantly improving the response accuracy or relevance, making it a convenient but superficial alternative to the RAG Model approach.

Fine Tuning

Fine-Tuning is where the parameters (adjustable values that optimise an LLM's learning) of a LLM are altered to fit a specific task, often to optimise how the LLM handles a new knowledge base (Chawre, 2023). Fine-Tuning adapts a model to specific tasks whereas a RAG model enhances responses by drawing upon external data. The fundamental approaches to Fine-Tuning an LLM are either where only the final layers of the model are trained on task specific data, such as Feature Extraction, and Full Fine Tuning, where the entire model is trained on the specific data (Chawre, 2023). If this approach had been implemented for this research, Feature Extraction would have been chosen as it specialises in small datasets with limited resources, whereas Full Fine-Tuning is used when there is a large dataset that is similar to the pre-trained original model dataset (Bosco, 2024).

The parameters are continuously adjusted to improve LLM outputs before deployment (Deepchecks, 2024). There are two main branches of this. Supervised Fine-Tuning (Table 5), where a pre-trained model is adapted to a specific task (Martinez, 2023) and Reinforcement Learning, which evaluates the model outputs through human or machine evaluators (Murel and Kavalakoglu, 2024) (Table 6).

Table 5: Supervised Fine-Tuning Techniques (data from Chawre, 2023)

Supervised Fine Tuning Technique	Description
Basic Hyperparameter Tuning	Manually adjusting the model parameters, such as the learning rate.
Transfer Learning	A pre-trained model is then fine-tuned on task specific data.
Multi-Task Learning	The model is tuned on different tasks simultaneously to identify any deviation or differences.
Few-Shot Learning	Giving the model only a few examples if the data is scarce or expensive.
Task-Specific Fine Tuning	Furthering training the model on a smaller dataset, changing the model to specific requirements.

Table 6: Reinforcement Learning Fine Tuning Techniques (data from Chawre, 2023)

Reinforcement Learning Fine Tuning Technique	Description
Reward Modelling	The model generates outputs with evaluators ranking outputs.
Proximal Policy Optimization	An algorithm that increases the feedback signals that guide the model towards better performance.
Comparative Ranking	Learns from and compares evaluator rankings of multiple outputs to adjust future responses.
Preference learning	Evaluators show their preference between pairs of outputs, the model then adjusts future responses.
Parameter Efficient Fine Tuning	Modifies only a subset of LLM parameters.

There are advantages to implementing a Fine-Tuning approach. Fine-Tuning can enhance LLM outputs. For example, Fine-Tuning increased Phi-2's, a large language model developed by Microsoft accuracy on financial analysis from 34% to 85% (Hughes, 2023). Fine-Tuning can also significantly reduce costs depending on model and dataset size. GPT 3.5 (OpenAI) charges \$0.50 per one million tokens (Iterations, 2025), a relatively small amount compared to other alternatives such as LLM creation.

Nevertheless, Fine-Tuning does have disadvantages which diminish its feasibility. The inputted dataset is static so it does not update with new information, meaning that it requires frequent re-training or risk accessing outdated information. The Gazetteer of Scotland (Gittings, 2023), the data source for GeoBot, is continuously updating which means that re-training would be often needed. One of the biggest negatives of Fine-Tuning is also the phenomenon of catastrophic forgetting (Greyling, 2024), where a model

forgets previously learned information (such as the original LLM training data) as new knowledge is inputted. GeoBot could then be unable to hold a conversation or perform simple tasks such as coordinate retrieval from the database.

When Fine-Tuning an LLM the data can be used internally, unlike a RAG Model which data is stored in an external database. However, The LLM's can then memorise and share this sensitive training data (Yan *et al*, 2025), detrimental to data security. For example, Samsung Engineers shared corporate data with OpenAI's ChatGPT which was then used as training data, potentially meaning that the sensitive data could be retrieved with specific prompts (Ray, 2023). Criminals can also exploit the LLM architecture and access sensitive data through backdoor attacks (Du *et al*, 2024). The Gazetteer for Scotland, while not containing sensitive data, does have copyright and cannot be shared without credit given. The amount of repeated Fine-Tuning required, combined with data privacy risks, means that this approach was not chosen for creating GeoBot.

LLM Model Creation

This is where an entire LLM Model is created. For GeoBot, this would mean creating an LLM Model which is purely trained on the Gazetteer for Scotland. While this is technically a valid approach to improve LLM Scottish geographical responses, this was not chosen as creating an entirely new LLM model is an extensive process, involving a high level of technical expertise. The entire Transformer Architecture must be built (Naminas, 2024), the foundation architecture of most modern LLM's, the process of which is seen in Figure 3.

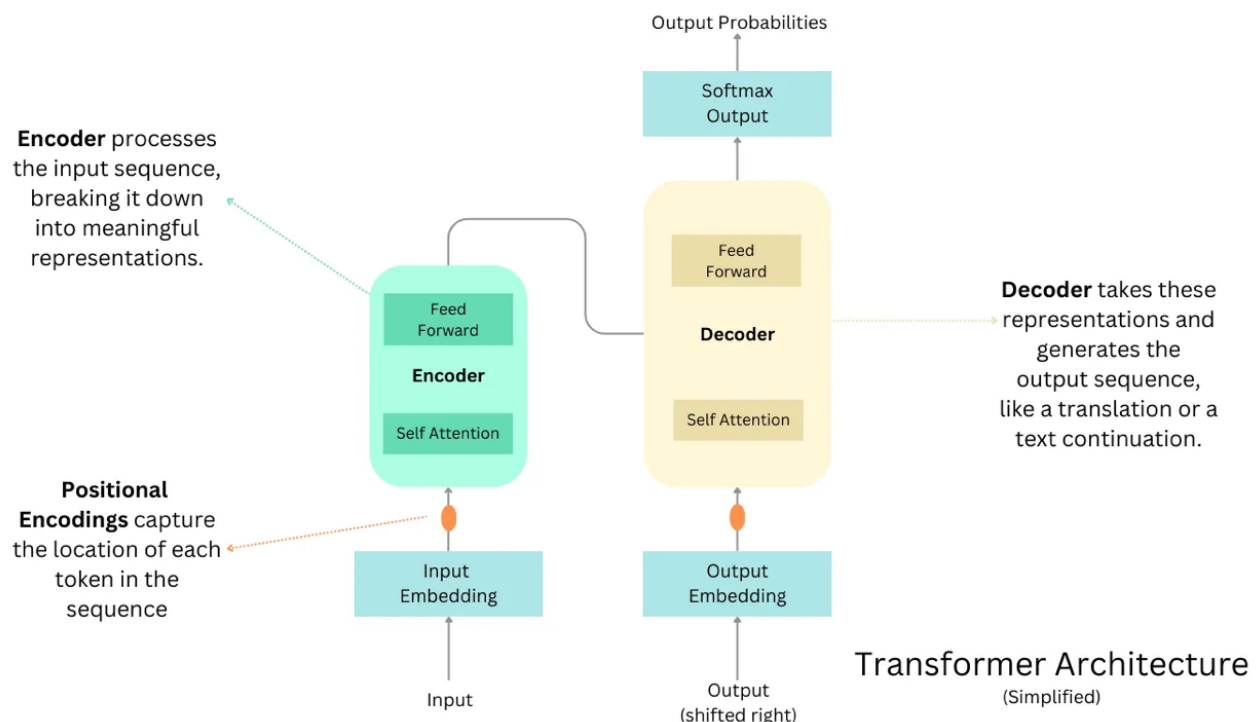


Figure 3: Transformer Architecture (Gal, 2023)

This approach was also not chosen due to its high cost, although this can vary widely depending on the training data requirements and the amount of memory space needed (Spheron, 2024). Small LLM's rely on less parameters (variables that are learned from training data (Belcic and Stryker, 2025), reducing training costs and memory space. The Phi-3 small language models, developed by Microsoft (Beatty, 2024), draw upon smaller datasets from quality documents such as textbooks, instead of a wealth of text data. This results in a suite of models that have under 14 billion parameters (compared to GPT-4's rumoured 1.76 trillion parameters (Lubbad, 2023)) and are well-suited for tasks that don't require extensive reasoning (Beatty, 2024). However, producing this amount of parameters is still beyond the aims of GeoBot and to produce a larger LLM, such as GPT-4, the cost becomes insurmountable (OpenAI spent between \$80-100 million training GPT-4 (Williams, 2024)). Once the LLM has been constructed, the running costs are also extreme; OpenAI also spends around \$4 billion to run ChatGPT (Rosenburgh, 2024) with data centres located across the globe.

However, there are still advantages to this approach. Despite being extremely complex, the limitless customisation capabilities could theoretically enable significant improvement to Scottish geographical responses if designed appropriately and there are no running charges from relying on an external LLM's architecture. Nevertheless, the cost of creating a new model is insurmountable for this research. Despite GeoBot needing much less storage and processing power than a commercial LLM, the extreme cost makes this option not viable.

For GeoBot to improve Scottish geographical responses, several LLM optimization techniques were considered. Direct LLM input and Prompt Engineering, although simple and cost-effective, lacked the depth needed to address data inaccuracies while Fine-Tuning offered improved accuracy but this was at the detriment of frequent retraining and privacy risks. Furthermore, creating a Custom LLM provided extensive customization but was prohibitively expensive. Therefore, ultimately the RAG model approach was selected for GeoBot as it can use updated external data whilst providing high data security and being cost effective, making it the most viable choice for GeoBot.

Alternatives to a Python coded RAG Model

In addition to choosing a general RAG Model methodology, the precise RAG Model methodology was also considered. While the chosen Python coded RAG model methodology was chosen to its high customisability, No-Code chatbots and Cloud Service Providers could have also been utilised to host and deploy GeoBot.

Cloud Service Providers

A Cloud Service Provider is a third-party company offering cloud computing storage and application services (Microsoft Azure, n.d). Google Cloud Platform (GCP) and Amazon Web Services, two Cloud Service Providers, both offer RAG model capabilities. GCP hosts Vertex AI, specifically Vertex AI RAG Engine, which provides a pre-built RAG methodology. Figure 4 shows a workflow of a website integrated RAG chatbot using GCP. Similarly, Amazon Web Services hosts Amazon Bedrock, a service that uses LLM's to automate the complete RAG workflow, shown in Figure 5.

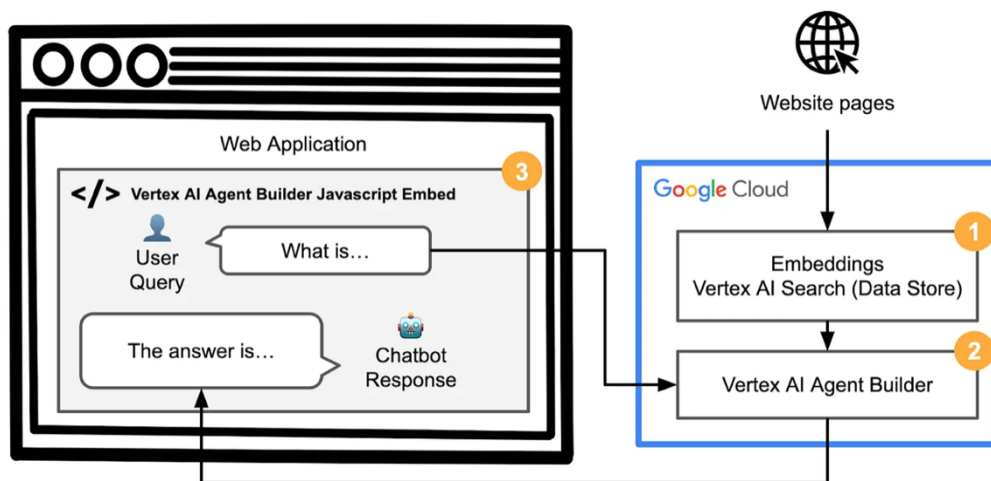


Figure 4: GCP RAG Model Chatbot Workflow (DiDonato, 2024)

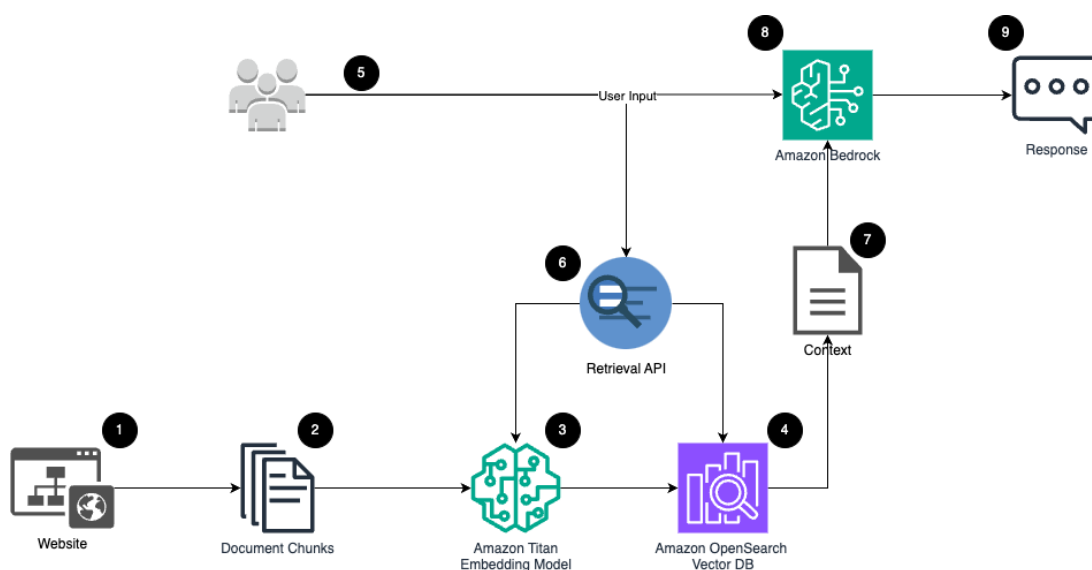


Figure 5: Amazon Web Services Bedrock RAG Workflow (Saeed, 2024)

Employing a cloud-based methodology does have advantages. Richards (2024) states that Cloud Platforms provide data security through encryption, as well as providing flexible and scalable data storage resources. GCP and AWS both also offer a pay as you go model (Hafnar, 2024). However, this cost can quickly rise if the amount of database data increases.

However, this methodology was ultimately not chosen to a relative lack of customisation compared to a Python Coded RAG Model. Table 7 shows a pairwise comparison of the two discussed Cloud Service Platforms (GCP and AWS) and the Python RAG Model which utilises LLM Model OpenAI's GPT-4. While this is subjective evaluation, it does show a preference towards a Python coded RAG Model due to customisation and scalability. While the methodology is less complex for GCP and AWS due to the automated workflows, purely using OpenAI's API Key in a Python coded RAG Model enables complete customization through the creation of python scripts, instead of relying on a pre-determined workflow hosted on GCP or AWS. Due to this, a Cloud Service Platform option was not chosen for GeoBot.

Table 7: Pairwise Comparison. 1 = first Model preferred, 0.5 = equal, 0 = second Model preferred

Criteria	GCP vs AWS	GCP vs Python RAG Model (PRM)	AWS vs Python RAG Model (PRM)
Methodology Complexity	0.5 (Equal)	1 (GCP preferred)	1 (AWS preferred)
Pricing	0.5	0.5 (Equal)	0.5 (Equal)
Customisation	1(GCP preferred)	0 (PRM Preferred)	0 (PRM preferred)
Environmental Impact	0.5	0.5	0.5
Scalability	0.5	0	0
Platform	Total Score		
GCP	5		
AWS	4		
OpenAI	6		

No-Code Chatbots

In addition to implementing a No-Code chatbot for the comparative chatbot Botsonic, this approach was also considered for GeoBot itself. A No-Code chatbot is an AI-powered virtual assistant that uses a customisable interface or workflow to design the chatbot instead of utilising programming logic (PlayAI, 2025). While a No-Code chatbot is not strictly a RAG Model, it implements a RAG model approach through implementing information from external sources and so was considered. This allows users to deploy chatbots without writing any code, a simplified and faster development process than coding a Python RAG Model. Table 8 evidences a wealth of no code chatbots, ranked based on criteria from Woyera (2024). All these no-code chatbots show proficiency in customisation, ease of use and data integration, with Botsonic (the platform chosen for the comparative chatbot) scoring the highest. However, none of these offered the customisation capabilities (for example map integration, customisation of GeoBot's response style) that coding a RAG model enabled. For this lack of customisation, this approach was not chosen for GeoBot.

Table 8: No Code Platform Chatbot Rankings (Information from Woyera, 2024)

Rank	Platform	Ease of Use	Customisation	Data Integration
1	Botsonic	9/10	GPT-3.5 for free users, paid users use “GPT Router”, lets you use multiple AI models	Diverse (PDFs, Docs, URLs)
2	ChatBase	9/10	Multiple OpenAI Models	High (400k character limit)
3	Wotnot	8/10	Variety of OpenAI models	URL scraping or file upload of txt, csv, pdf or docx, no files exceeding 5MB, all files must total less than 100k tokens
4	Tidio	7/10	Anthropics Claude LLM	Website URL, Manual Q and A, File Imports for paid members only
5	Landbot	6/10	Only uses 3.5 OpenAI Model	Limited (50,000 character limit)
6	Botsify	7/10	Multiple OpenAI Models	Crawls website URL's, uploads documents: pdf, docx, txt
7	Juji	5/10	No choices of AI models to choose from	Allows users to upload different documents, pdf, txt, doc, xlsx, and csv. Websites via a URL link

Alternative LLM Models

Despite OpenAI's GPT-4 being selected, there are a wealth of other LLM's available that would be suitable for GeoBot. The main LLMs considered were Gemini and Claude. Developed by Google, Gemini (Google, 2023) is an interface to a LLM which can process and output text, images and audio (multimodal). Similarly, Claude (Anthropic, n.d) is a suite of multimodal LLM's developed by Anthropic and emphasises AI safety, privacy, ethics, despite being closed source (Proudfoot, 2024).

GPT-4, Gemini and Claude all perform consistently when simulating human understanding. Table 9 shows that Gemini and GPT-4 both score highly in MMLU reasoning, the evaluation of a language model's ability to perform logical reasoning, with Claude scoring lower at 88.3%.

However, Gemini does consistently outperform both GPT-4 and Claude. Pichai and Hassabis (2023) demonstrate that Gemini Ultra, the most expensive Google LLM at \$250 a month (Ben-Yair, 2025), outperforms GPT-4 in general reasoning capabilities, scoring 3.6% higher in a MMLU (Massive Multitask Language Understanding) reasoning test (Table 9) (Pichai and Hassabis, 2023). Gemini also outperforms GPT-4 and Claude on reasoning and knowledge through the GPQA test, a challenging dataset of 448 multiple-choice questions written by experts in biology, physics, and chemistry (Rein *et al*, 2023). Gemini does seem the most logical choice for GeoBot due to its high performance, however due to privacy concerns OpenAI's GPT-4 was ultimately chosen. If the RAG Model was using non copyrighted data, Gemini would be utilised due to its superior performance.

Table 9: LLM Comparison (Information from Meta (2025))

	Claude Sonnet 4	Gemini	GPT-4
Cost per 1 Million Tokens (Meta, n.d)	\$3	\$0.17	\$4.38
GPQA Diamond (Reasoning and Knowledge)	59.4	60.1	53.6
MMLU General Reasoning Test	88.3% (Anthropic, n.d)	90% (Pichai and Hassabis, 2023)	90.2% (Buntz, 2025)

Chapter Summary

This chapter highlights a RAG Model's ability to enhance Scottish geographical responses by integrating external data. Alternatives like Direct LLM Input, Prompt Engineering, Fine-Tuning and LLM Model creation were considered but ultimately disregarded due to privacy risks and high costs. More specifically, a Python-coded RAG Model was chosen over Cloud Platforms or No-Code chatbots because of greater customization and flexibility, with OpenAI GPT-4 selected as the LLM due to its high data security through ELM, despite Gemini's stronger performance.

Chapter 3: GeoBot's Logical Architecture Overview

GeoBot's logical architecture is constructed of key components such as data selection and processing, user interaction and response generation. GeoBot is a Retrieval Augmented Generation (RAG) model that utilises OpenAI's GPT-4 Large Language Model and draws upon data from the Gazetteer for Scotland in a PostgreSQL database (an open-source, object-relational database system), also integrating an interactive map feature. This chapter is split into two main sections: the Frontend and Backend GeoBot architecture. The Frontend is the part of a website that users directly interact with, and the Backend is the part that is not directly accessed by the user (Code Academy, n.d).

Figure 6 shows the full workflow of GeoBot, from the user entering a query to the GeoBot response being displayed on the website Frontend. The Backend consists of data collection and storage, which is processed and used to generate a response through a Flask app. A Flask app is a web application framework and was chosen as it is essential for connecting python applications to a website (Djouri and Walia, 2024).

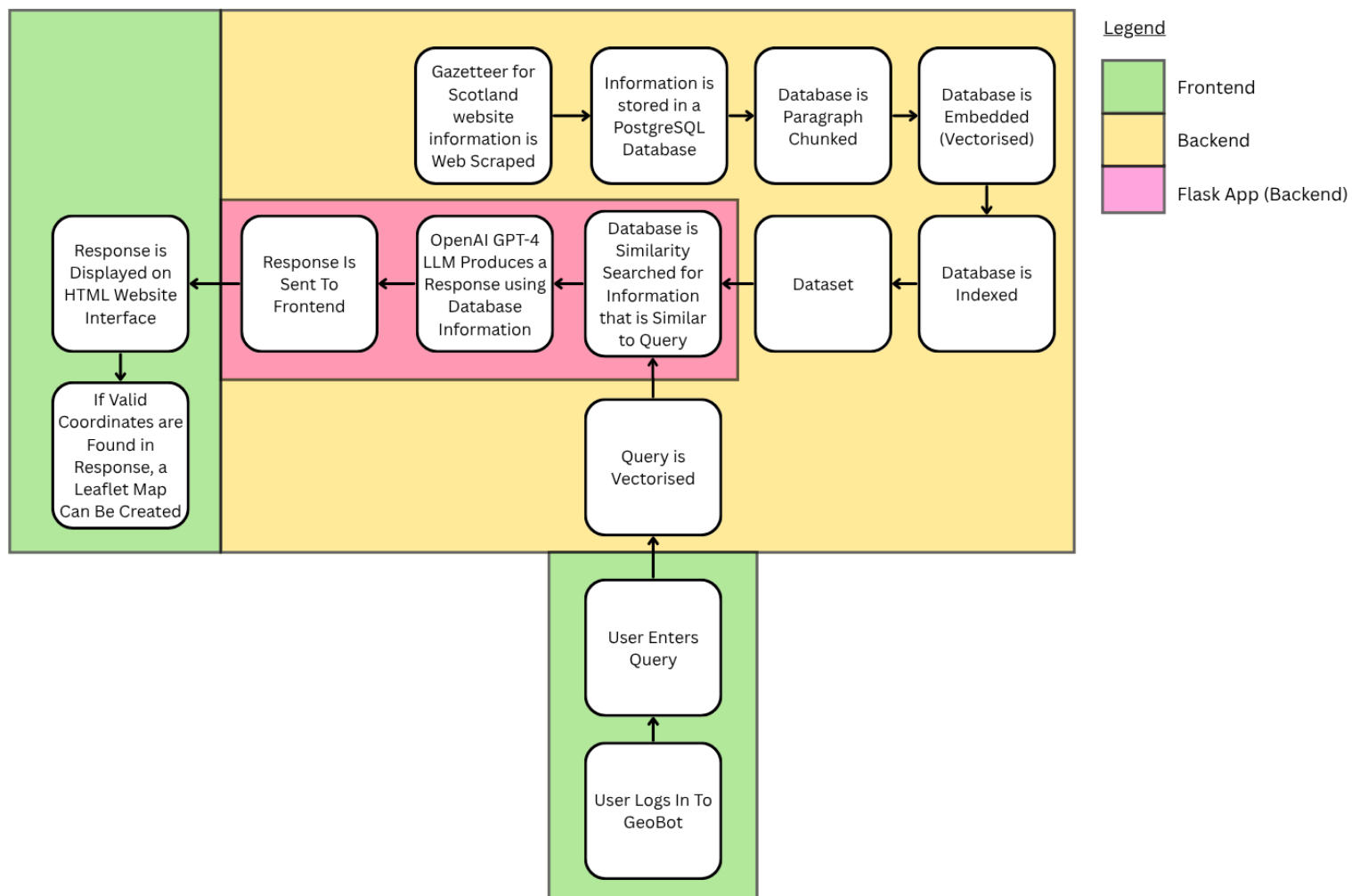


Figure 6: GeoBot Full Process Workflow

Figure 7 shows the file structure of GeoBot. This file structure was designed for clarity, scalability and maintainability to separate the processing scripts, the Flask app and the website templates. While the alternative, to place all Python scripts in one folder, is viable, this is not as readable and is confusing when adding further scripts and functionality.

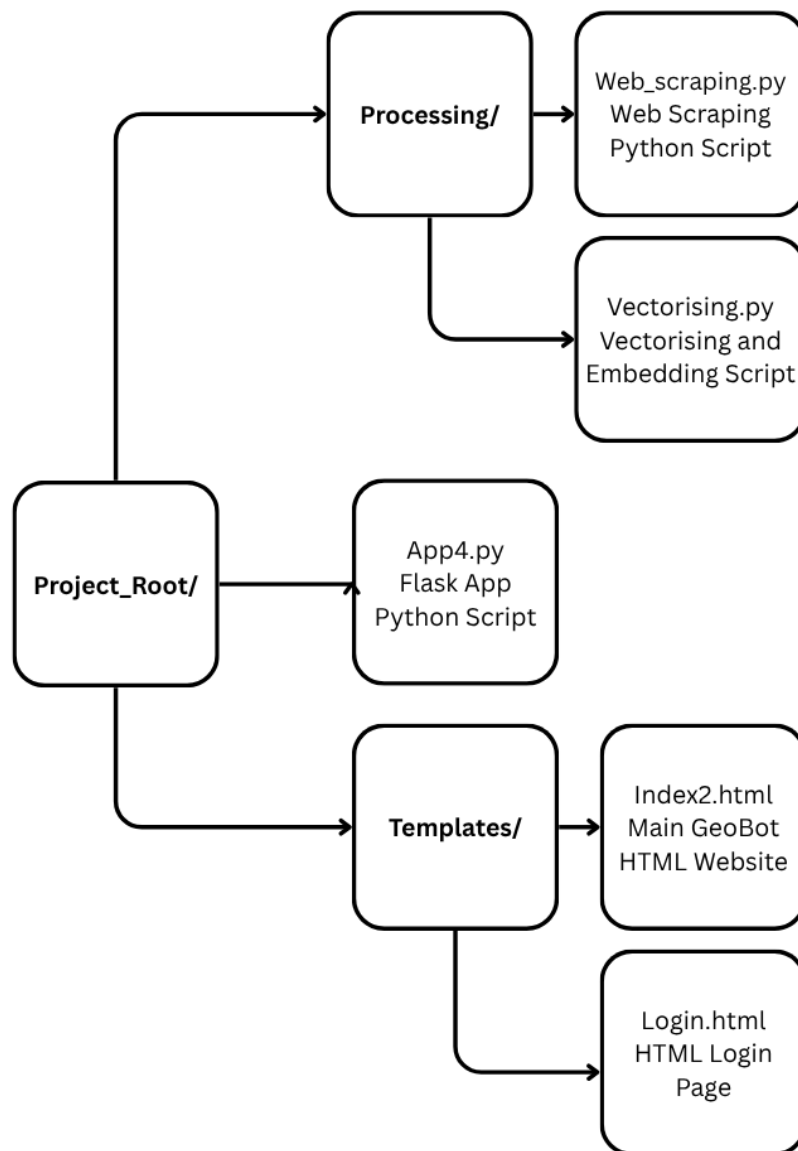


Figure 7: GeoBot File Structure

Chapter 4: GeoBot's Backend Architecture

This section consists of two main parts: the database (data selection, data storage and data processing) and the Flask app methodology. The chosen methodology for each process will be discussed, with weaknesses and improvements acknowledged, as well as alternative methodologies presented.

GeoBot's Database Methodology

Data Selection

Gazetteer for Scotland Justification

The goal of data selection was to enhance GeoBot's capabilities by integrating a comprehensive database of Scottish geography to provide accurate geographical information. Inaccurate or outdated data can substantially decrease the quality of a RAG Model's responses (Barnett *et al*, 2024). The Gazetteer for Scotland (Gittings, 2023) is a continuously updating comprehensive database of Scottish geography, with 27858 entries on subjects such as tourist attractions, industry data and historical sites. Figure 8 shows the comprehensive main webpage for the Haymarket entry (Edinburgh district). This dataset was chosen as it is the first comprehensive Gazetteer produced for Scotland since 1885 (Gittings, 2023) and contains viable coordinate data for the GeoBot map feature. However, a weakness of the Gazetteer for Scotland is that only using one singular dataset, written by limited authors, could unwittingly cause bias or preference to certain Scottish locations. This bias could be remedied by the implementation of additional datasets, seen in Table 10.

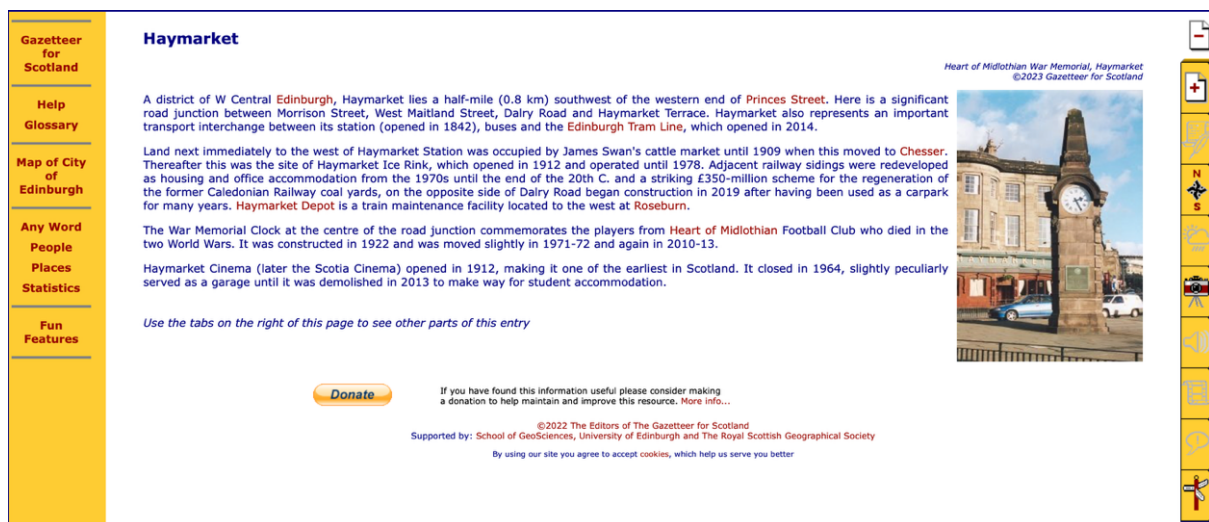


Figure 8: Haymarket Scotland for Gazetteer entry (Gittings, 2023)

Alternative Data Sources

Despite the Gazetteer for Scotland dataset being chosen, there are other datasets that could have been utilised, shown in Table 10.

Table 10: Alternative Data Selection Methodologies

Data Source	Description	Reason For Data Inclusion/Exclusion	Potential Utilisation
The National Records of Scotland (n.d)	The National Records of Scotland provides detailed council area profiles, such as population estimates, migration and life expectancy.	Does not contain comprehensive details on Scottish geographical context or historical sites, unlike The Gazetteer.	Could be added to GeoBot to add demographic data.
The Scotland's Environment Agency (n.d)	The Scotland's Environment Agency holds a wealth of environmental data, including Met Office climate data and heritage data from Historical Environment Scotland.	Does not contain comprehensive details on Scottish geographical context or historical sites, unlike The Gazetteer.	Could be added to GeoBot to provide environmental data to GeoBot responses.

Web Scraping The Gazetteer for Scotland Data

Web Scraping Justification

Once the data source had been selected, the information was Web Scraped into the PostgreSQL database (Appendix A). Web Scraping is the process of extracting information from a website and storing for later use (National Library of Medicine, 2022). Web Scraping was chosen as it is a powerful tool for data acquisition, especially when supplying datasets to machine learning models, such as LLM's (Chaudhary, 2024). The objective of this approach was to extract only location data from the Gazetteer for Scotland focusing on Edinburgh greenspaces for a proof of concept. Because of this, efficient navigation of multiple subpages was required. This specific greenspace dataset was chosen to produce a targeted and understandable chatbot for user testing and results evaluation within the time constraints of this research.

Despite Web Scraping enabling targeted website information retrieval, one significant disadvantage is the associated privacy concerns. For example, there has been widespread concern with OpenAI's web scraper GPTBot gathering a wealth of data to train the LLM's, only stopping short of collecting information with personally identifiable information. The New York Times has also alleged that ChatGPT unlawfully generates outputs using its copyrighted materials (Tiedrich, 2024). While for this research the owner of the Gazetteer for Scotland granted permission to collect this data, caution must be taken if replicating this methodology with other websites and datasets.

Selenium Web Scraping Justification

The Web Scraping tool Selenium was chosen for the Gazetteer for Scotland webpages. This tool automatically navigates a web browser to extract information and was chosen as it is open source and has powerful automation capabilities (Heath, 2023). This Web Scraping process was executed using a Python script, the workflow of which is shown in Figure 9. The Selenium web driver searches for the website paths which are defined in the Python script, with the relevant information being automatically stored in a PostgreSQL database.

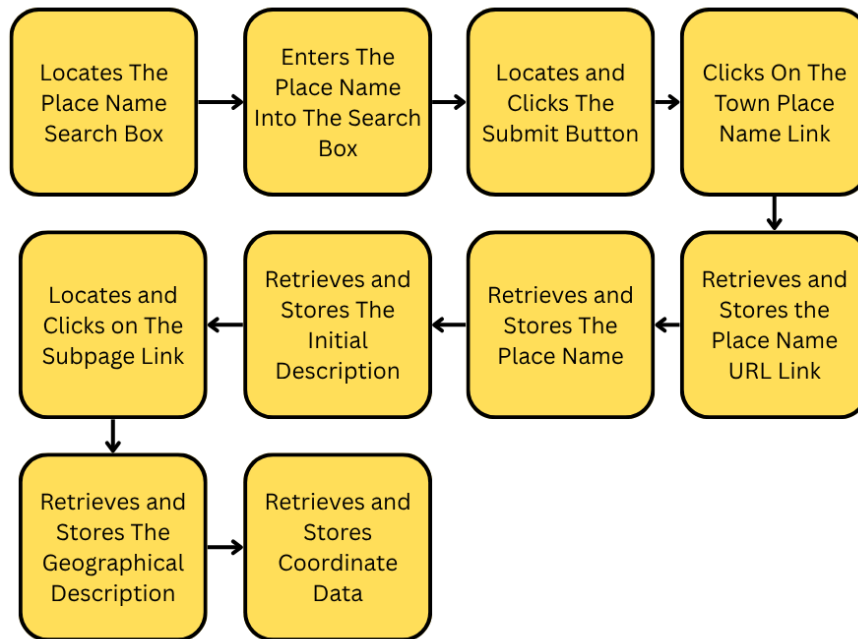


Figure 9: Selenium Web Scraping Workflow

A limitation of the chosen Selenium approach was that only a small section of the Gazetteer website was Web Scraped and the Python script had to be manually altered for each place and greenspace. Additional Web Scraped information would have improved GeoBot's geographical responses. Figure 9 focuses on one singular webpage, but for wide scale web scraping this script should become more expansive. Figure 10 shows the methodology of an improved Web Scraping approach through the Edinburgh webpage, where the other areas of interest, 'attractions', are also collected through a Selenium loop. Ideally, the entire Gazetteer for Scotland database would be Web Scraped. However, this is an extremely complex approach, requiring multiple Selenium loops, and is somewhat redundant for this research when considering the existing Gazetteer for Scotland database that the Gazetteer already draws from.

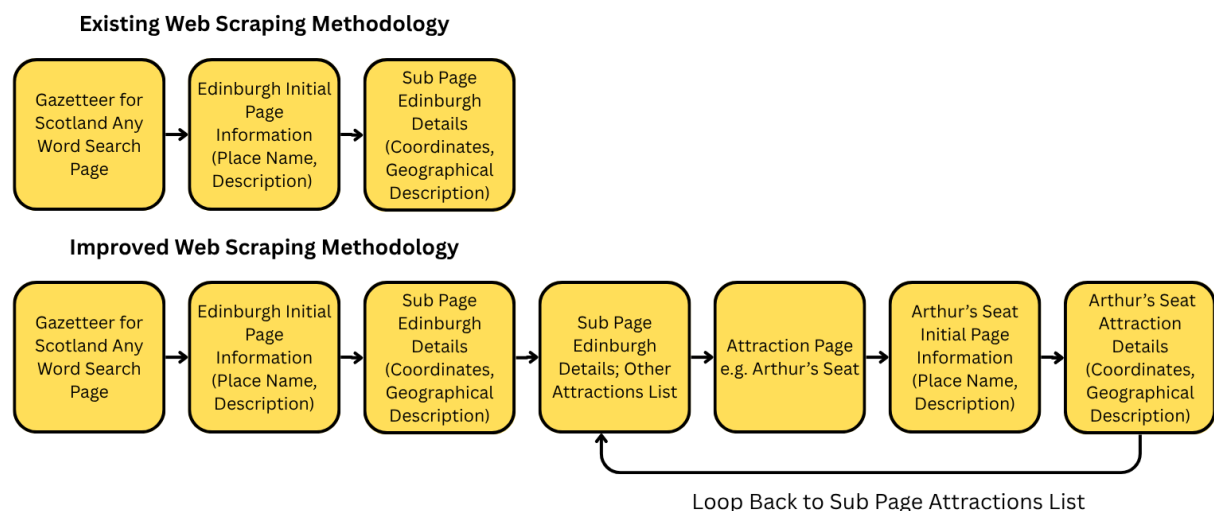


Figure 10: Web Scraping Original and Suggested Improved Web Scraping Methodology

Alternative Web Scraping Methodologies

The other options for Web Scraping, shown in Table 11, were also evaluated.

Table 11: Main Web Scraping Tools (Information from Heath, 2023)

Web Scraper	Description	Advantages	Disadvantages
Beautiful Soup	Python parsing library that retrieves data from HTML files by creating a parsing tree to navigate to specific elements in a webpage.	Easy to use interface and Open source library.	Slow Processing. Not recommended for complex web scraping tasks.
Scrapy	Python based framework that has web crawling, data extraction, and processing functionality.	Fast processing and customisable.	Steep learning curve and not suitable for simple tasks.
Selenium	Library that automates web browser to extract information.	Open source and has powerful automation capabilities.	Slower than other options.
Octoparse	No code tool using a point and click interface.	User-friendly interface.	Limited automation capabilities.

In particular, Octoparse (n.d) was considered due to it's no code user friendly interface. Octoparse automatically scans a URL and selects text, organising this into an interactive table which can be easily exported. This works well for simple websites where there is limited interactivity, however for websites with multiple subpages, such as the Gazetteer, this is not suitable as Octoparse, when tested, failed to automatically recognise and navigate to the subpage links. This navigation is essential as the subpages contain further description and coordinate data. Therefore, because of the complexity of the Gazetteer for Scotland websites, Selenium was chosen.

Data Storage

PostgreSQL Database Justification

Once the web page data has been collected, it needs to be stored (Appendix A). Storing the Web Scraped data is crucial as it enables successful information retrieval by GeoBot. PostgreSQL, an open-source object-relational database, was chosen for its compatibility with complex data types and geospatial support through extensions like PGvector (PostgreSQL, n.d), essential for handling coordinate data. Table 12 shows the PostgreSQL table structure once the web scraped information has been inserted, with Figure 11 evidencing the commands used to create the GeoBot PostgreSQL database.

Table 12: GeoBot PostgreSQL Initial Table Structure

Column	Type	Nul l	Default	Example
Row_ID (Primary Key- unique identifier)	Integer	Not Nul l	Nextval (<code>'millie.chatbot _row_id_seq'::regclass</code>). Nextval(<code>'millie.chatbot_row_i d_seq'::regclass</code>) This means that the row_id number auto- increments.	1
Place_Na me	Text			Edinburgh
Latitude	Text			55.9523
Longitude	Text			-3.1882
Descriptio n	Text			Built on a group of hills, it is situated between the Firth of Forth...
Link	Text			https://www.scottish- places.info/towns/townfir st337.html

```

CREATE TABLE millie.chatbot (
    -- Stores the name of the place
    place_name text,
    -- Stores the latitude as text
    latitude text,
    -- Stores the longitude as text
    longitude text,
    -- Contains a descriptive text of the place
    description text,
    -- Contains URL link
    link text,
    -- Unique identifier for each row, automatically incremented
    row_id serial PRIMARY KEY,
    -- Embedding vector of the place name
    place_name_vector vector(1536),
    -- Embedding vector of latitude
    latitude_vector vector(1536),
    -- Embedding vector of longitude
    longitude_vector vector(1536),
    -- Embedding vector of the description
    description_vector vector(1536),
    -- Embedding vector of the link
    link_vector vector(1536)
);

```

Figure 11: PostgreSQL Table Creation Commands

The text classification for most columns was chosen due to its flexibility; a variety of data structures can be stored. However, there is arguably an overuse of the text classification and despite successful integration with GeoBot another classification, such as decimal or float, would have been more suitable for the latitude and longitude columns due to the complexity of the coordinate data.

Another limitation of this database was the small size. Due to the test dataset only composing of Newbigging in Angus and Edinburgh greenspace information, the current GeoBot dataset did not utilise multiple tables or foreign keys. However, this would be a beneficial improvement to the GeoBot database, especially if expanded, to ensure that the information was still effectively retrieved by the OpenAI LLM. Figures 12 and 13 show an example ER diagram and associated table creation SQL code for a more complex table structure, where there is one general table for towns and one table for the associated town attractions (such as greenspaces, museums and famous figures).

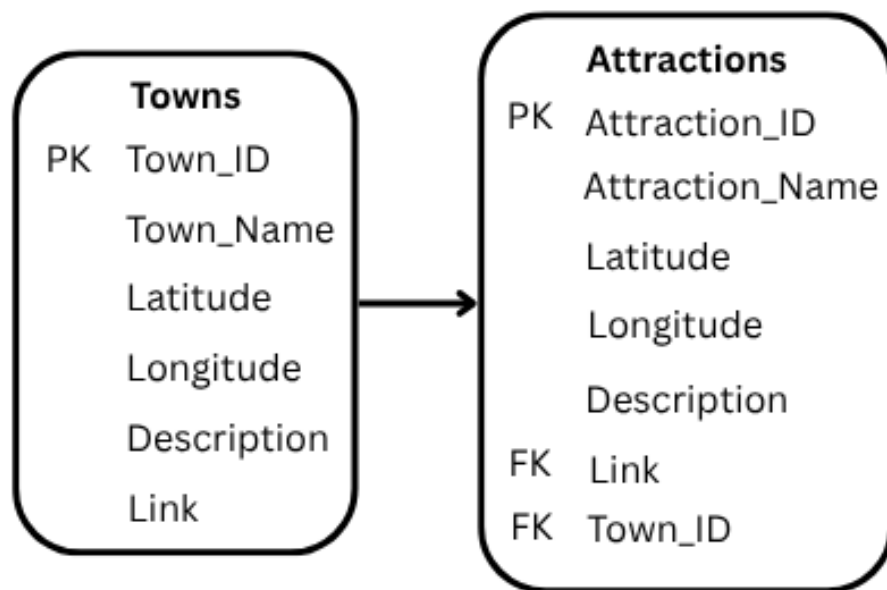


Figure 12: ER Diagram for proposed PostgreSQL tables. PK (Primary Key) and FK (Foreign Key)

PostgreSQL Table Creation

```

-- Creates a table for each town or village.
CREATE TABLE towns (
    town_id SERIAL PRIMARY KEY,
    town_name TEXT NOT NULL,
    latitude FLOAT,
    longitude FLOAT,
    description TEXT,
    link TEXT
);

-- Creates a table for attractions associated with each town or village.
CREATE TABLE attractions (
    attraction_id SERIAL PRIMARY KEY,
    attraction_name TEXT NOT NULL,
    latitude FLOAT,
    longitude FLOAT,
    description TEXT,
    link TEXT,
    town_id INTEGER,
    FOREIGN KEY (town_id) REFERENCES towns(town_id)
);
  
```

Figure 13: Proposed PostgreSQL Table SQL Creation Code

Alternative Data Storage Methodologies

Despite the PostgreSQL Database successfully storing the web scraped Gazetteer for Scotland Information, alternative methodologies were considered (Table 13).

Table 13: Data Storage Alternative Methodologies

Data Storage Method	Description	Reason For Inclusion/Exclusion
Gazetteer For Scotland Oracle Database	Oracle is a closed-source database management system (Oracle, 2014). The Gazetteer for Scotland original Oracle database could have been used. The biggest limitation of GeoBot is the small amount of data in the PostgreSQL database, reducing the number of queries that GeoBot can successfully reply to. The Oracle database would remedy this by enabling GeoBot to draw upon all of the Gazetteer for Scotland data, not just information on Edinburgh Greenspaces, enabling much richer responses.	This was not chosen for this research due to the prohibitive cost of embedding the entire database. Embedding converts text into numerical vectors and is essential for LLM's to understand the database data (Hernandez, 2023). The Gazetteer for Scotland contains 27858 location entries and is constantly being updated, compared to less than 50 entries in the GeoBot PostgreSQL database information. The cost to embed GeoBot's existing PostgreSQL database was less than approximately \$0.10 per entry (OpenAI, 2025). However, the Gazetteer has over 550 times more entries. This would perhaps total \$2,785.80 to embed the entire database, despite not accounting for varying entry lengths.
Elasticsearch	This is an open source, distributed search and analytics engine (Elasticsearch, n.d). This was considered as it completely automates the Data retrieval, storage and processing component of GeoBot (Figure 14), massively reducing methodology complexity. Elasticsearch's 'Playground' interface also enables the building and deployment of the RAG interface, instead of building GeoBot's custom Python RAG Model separately, on top of the Elasticsearch database.	This approach was not chosen as it was unfortunately too resource intensive to be supported on the host computer system. It is a very viable option and if suitable would have been chosen instead of the Selenium Web Scraping and PostgreSQL data storage.

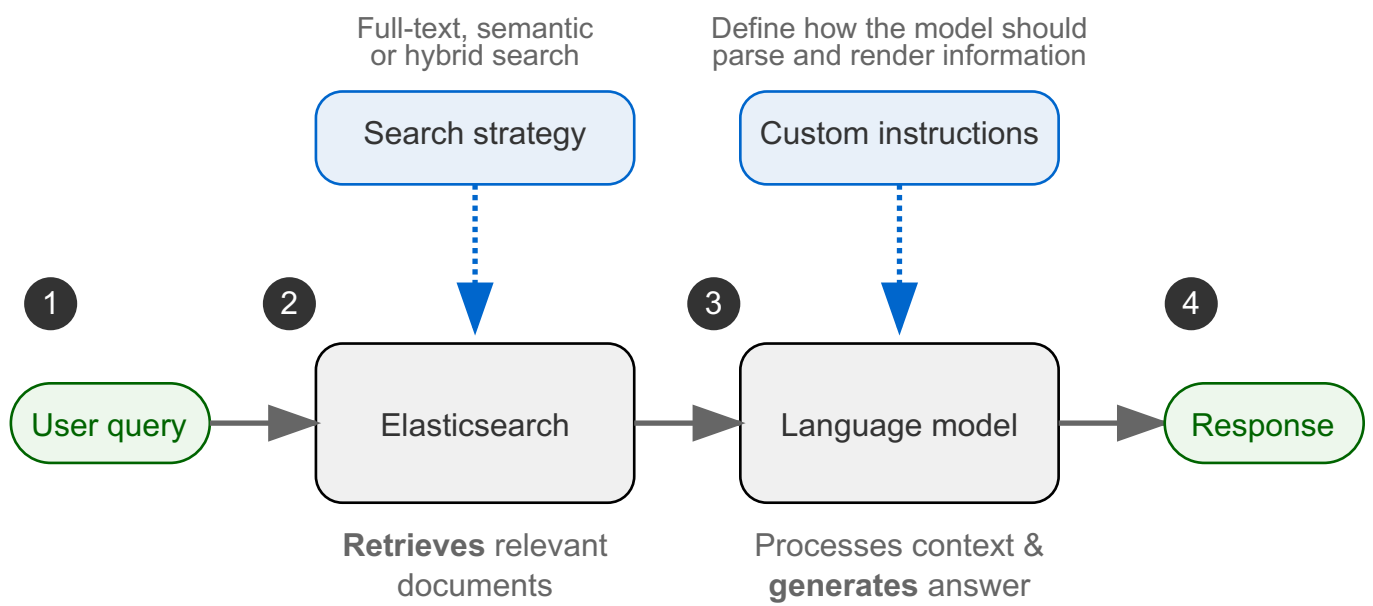


Figure 14: Elasticsearch RAG Workflow (Elasticsearch, n.d)

Data Processing

Data Processing Justification

GeoBot's data processing methodology consists of Chunking, Embedding and Indexing the PostgreSQL database. This was chosen to enhance the data retrieval and accuracy of the OpenAI LLM used In GeoBot. Chunking divides the data into smaller sections (Kutumbe, 2024) so that the data included in the response is specific to the user query, instead of retrieving the entire database for every query (Donovan, 2024). Embedding then converts the data into numerical vectors (a number representation of a word) (Hernandez, 2023) so the LLM can read the data, with Indexing finally creating a data structure on the vector PostgreSQL database columns to ensure fast and efficient data retrieval by the LLM (Yashwanth, 2025).

While the only essential component to this methodology is Embedding, without which the OpenAI LLM could not understand the database information, all components were selected to achieve a higher overall efficiency and data retrieval accuracy. Chunking breaks down information into manageable parts, increasing the relevance of the information that is returned, and Indexing optimizes search speed, making it easier for the OpenAI LLM to provide precise responses based on user queries.

Chunking Data

Paragraph Chunking Justification

There are a variety of Chunking methods available, shown in Table 14. Paragraph Based Chunking was chosen as it is one of the least complex approaches and most closely matches the text structure of the Gazetteer for Scotland entries (Mastering LLM, 2024) so that the meaning of the entry is not lost. Paragraph Chunking also ensures that the information about features is not fragmented, the methodology of which is shown in Figure 15.

Table 14: Chunking Methods in order of complexity (Mishra (2024) and Schwaber-Cohen (2023))

Method	Description
(Least Complex) Fixed Size Chunking	Dividing text based on a pre-determined length.
Sentence Based Chunking	Divides text by each sentence.
Paragraph Based Chunking	Divides based on paragraphs.
Document Based Chunking	Divides documents based on its structure.
Semantic Chunking	Uses machine learning to divide text based on meaning.
Recursive Chunking	Continuously breaks down text based on delimiters such as subheadings.
Agentic Chunking	Uses LLM to determine how to chunk the text.
Modality-Specific Chunking	Divides depending on text type e.g. image, text.
Context-Enriched Chunking	Adds a metadata summary to each chunk to maintain text context.
(Most Complex) Hybrid Chunking	Combines multiple chunking methods.

```
-- Creates a backup table
CREATE TABLE millie.chatbot_backup5 AS TABLE millie.chatbot;

-- Clears the content the original table millie.chatbot
TRUNCATE TABLE millie.chatbot;

-- States a Common Table Expression (CTE) named 'split_descriptions'. Selects the relevant
columns from the PostgreSQL table millie.chatbot backup table.
WITH split_descriptions AS (
    SELECT
        row_id,
        place_name,
        longitude,
        latitude,
        link

-- Uses the PostgreSQL expression 'regexp_split_to_array' to split the description field into
an array of strings, seperating the text at the end of each paragraph('\n\n'). The Unnest
function then transforms this array into multiple rows, each containing one paragraph but
retaining the other row content (links, row_id, place_name). The coordinates and place name
are repeated for each row.

        unnest(regexp_split_to_array(description, E'\n\n')) AS paragraph

    FROM millie.chatbot_backup5
)

-- Inserts these new paragraph rows into millie.chatbot
INSERT INTO millie.chatbot (row_id, place_name, longitude, latitude, link, description)
SELECT
    row_id,
    place_name,
    longitude,
    latitude,
    link,
    paragraph
FROM split_descriptions;
```

Figure 15: GeoBot SQL Chunking Code

Alternative Chunking Methodologies

Despite Paragraph Chunking being chosen, there are several other chunking options shown in Table 14. Sentence Based Chunking was considered as an alternative, but was disregarded as some of the sentences do not explicitly mention locations and features, such as the last sentence in Figure 16 ‘This was removed in 2014 to undergo a lengthy and expansive restoration...’, which refers to the Whale Bone Arch but is not explicitly mentioned, meaning that the OpenAI LLM would not classify this sentence as belonging to a specific feature. For this research, the more complex chunking options such as Context-Enriched Chunking and Modality-Specific Chunking were deemed unnecessary due to the small amount of data in the PostgreSQL database.

The Meadows

One of the Mason's Pillars at the west end of the Meadows
©2025 Gazetteer for Scotland

Located on the **South Side** of **Edinburgh**, behind **George Square** and the former Royal Infirmary buildings on Lauriston Place the Meadows were designated as a Millennium Park in 2000. Together with the adjacent **Bruntsfield Links**, they provide 14.5 ha (36 acres) of valuable greenspace in the city. The area was originally called Hope Park (hence Hope Park Terrace) after **Sir Thomas Hope of Rankellor** (c.1681 - 1771). Around 1740, Hope drained the Borough (Burgh) Loch, or South Loch, which had previously occupied the site. The South Loch had been a water supply for the City, while the Nor' Loch, which was later transformed into **Princes Street Gardens**, had been created for defensive purposes. Hope laid the Meadows out as parkland, with narrow drainage canals, a summer house, tree-lined walks and avenues, bisected by Melville Drive. **William Burnes** (1721-84), a gardener and the father of poet **Robert Burns** (1759-96), worked briefly on the project around 1749. On several occasions the area has been threatened with development. In 1832, along with **Holyrood Park**, the Meadows were considered as the site of a grand cemetery but both were ruled out due to the unsuitability of the soil. In 1879, landscape gardener Edward Kemp (1817-91) prevented Middle Meadow Walk being widened into a carriage drive and a railway was nearly driven beneath the Meadows in 1949. Perhaps most seriously, misguided planners almost succeeded in pushing a six-lane motorway through the area in the 1960s. However, today the Meadows are much as they were laid out in the 18th C, except for the addition of tennis courts, croquet lawns and football pitches. The East Meadows was the site of the first **Edinburgh** football derby between the **Heart of Midlothian** and Hibernian clubs on Christmas Day in 1875.

Massive pillars at the east end of Melville Drive, topped by a lion and a unicorn, were built in 1881 and funded by the grateful Nelson brothers after the city fathers had quickly provided them with alternative buildings when their nearby printing works were struck by fire in 1878.

At the western end are more slender Mason's Pillars by **Sir James Gowans** (1821-90), which are topped by unicorns. These were built for the Edinburgh International Exhibition of 1886, which was held in an immense exhibition hall built on the Meadows. **Gowans** built the pillars using specimen stone from a number of different Scottish quarries. The name of the quarry was carved into each stone and, although several blocks have subsequently had to be replaced, they still provide an excellent record of the materials used on many of **Edinburgh's** fine buildings.

The unusual double whale's jaw-bone arch, which lies on the north side of Melville Drive opposite Marchmont Road, came from **Shetland** for the 'Zetland and Fair Isle Knitting Stand' at the International Exhibition. This was removed in 2014 to undergo a lengthy and expensive restoration, briefly reinstated in 2019 but soon removed again because of their delicate condition.



Figure 16: Excerpt from The Gazetteer for Scotland entry ‘The Meadows’ (Gittings, 2025)

Table 15 shows an example output of Paragraph Chunking. While this methodology was sufficient for this research, for true scalability another methodology such as Semantic Chunking would likely be beneficial for larger complex datasets. For this, the divides are calculated using the cosine distance between sequential embeddings (Martin-Short, 2024), so that the groups of text are distinct from each other. This would ensure that all chunks are contextually relevant without having to manually review.

Table 15: Example Chunking Output (Description Summarised)

Row_Id	Place Name	Latitude	Longitude	Description	Link
1	The Meadows	55.9412	-3.1923	Massive pillars at the east end of Melville Drive...	https://www.scottish-places.info/features/featurefirst990.html
2	The Meadows			At the western end are more slender Mason's Pillars...	https://www.scottish-places.info/features/featurefirst990.html
3	The Meadows			The unusual double whale's jaw-bone arch...	https://www.scottish-places.info/features/featurefirst990.html

Data Embedding

OpenAI's Text-Embedding-Ada-002 Model Methodology and Justification

Embedding is when text data is converted into vectors (numerical representations of words) (Hernandez, 2023). Using OpenAI's text-embedding-ada-002 model was essential as it allowed GeoBot to retrieve the most relevant database information, which could then be understood by GeoBot's LLM GPT-4.

Despite a minimal usage cost, OpenAI's Embedding model was chosen as it uses machine learning models to look at the context of the whole PostgreSQL database instead of just keyword matching. This achieves more accurate data retrieval than if GeoBot only used simple SQL database queries to retrieve the data (Li, 2025). Specifically, the text-embedding-ada-002 model was chosen as despite not being the most recent model released, it still performs strongly in efficiency and overall performance (Kutz, 2025).

Figure 17 showcases the general Embedding process, where data is converted to vectors and then plotted by the Embedder, with semantically similar words clustered together so the LLM knows where to retrieve them. Figure 18 shows how this process was applied to GeoBot (Appendix B), where each row of the PostgreSQL database was turned into vectors.

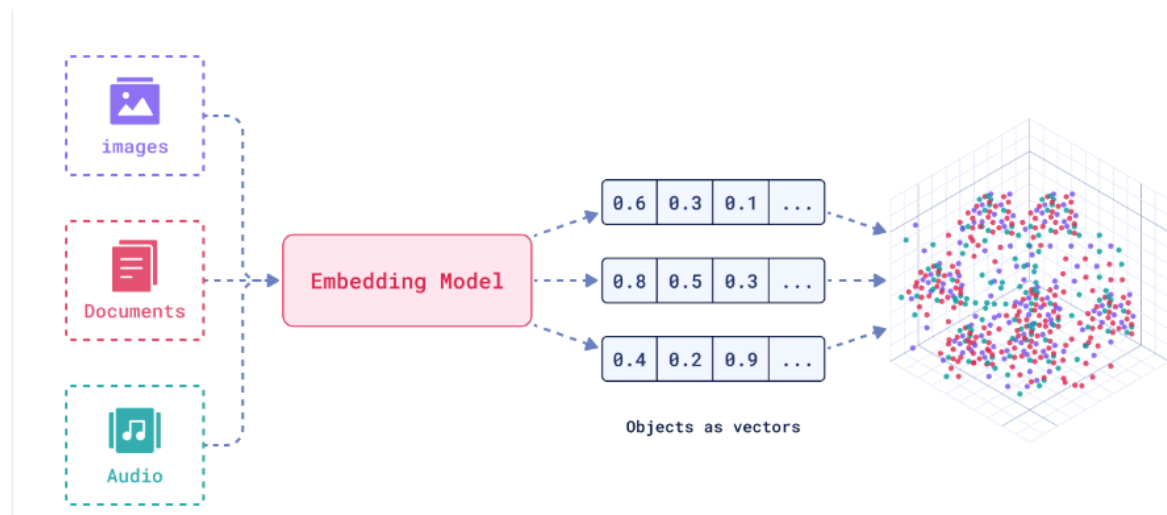


Figure 17: Embedding process (Ashutosh, 2024)

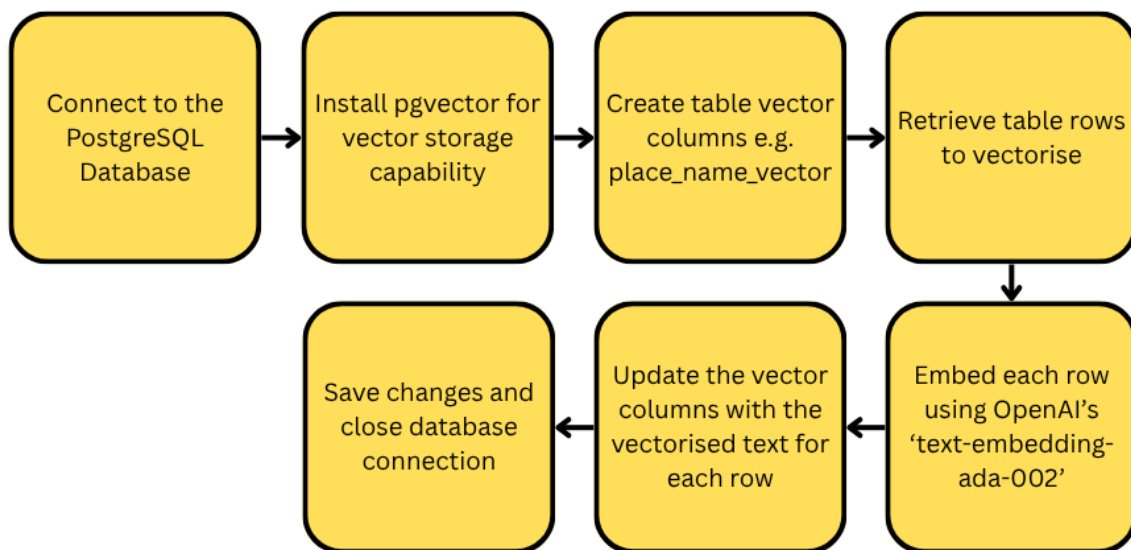


Figure 18: Embedding Python Script Methodology

Alternative Embedding Models

More advanced OpenAI Embedding models (Table 16), such as text-embedding-3-large, offer improved Embedding analysis and capacity for complex tasks. However, they incur significantly higher costs and so were deemed unsustainable for GeoBot. A limitation of the OpenAI Embedding models is that they are all closed source and cost a small amount per embedding. Hugging Face's free API for Embeddings (Espejel, 2022) was considered as an alternative to avoid Embedding processing costs. This would be viable but was ultimately not chosen due to the comparative streamlined ease of implementing the OpenAI Embedding model in GeoBot.

Table 16: OpenAI Embedding Model comparison (information from Kutz, 2025)

Models	Descriptions	Output Size	Computational Efficiency	Use Cases
Text Embedding-3-Large	This third generation embedding model offers the greatest capability for text, ideal for complex text analysis tasks.	3,072 dimensions	Lower (more resource-intensive)	Detailed tasks such as complex semantic analysis.
Text Embedding-3-Small	It is an enhanced version of the ADA model and third-generation model. This improves performance for faster processing.	1536 dimensions	Higher (less resource-intensive)	Simple keyword searches and quick text classification.
Text Embedding-ada-002	This second-generation embedding model exceeds the capability of 16 previous models for text embedding tasks.	1536 dimensions	Moderate (Balanced efficiency)	A wealth of tasks such as content recommendations and general text analysis.

Indexing

IVFFLAT Justification

Indexing creates a data structure on the vector PostgreSQL database columns so the data can be organised for retrieval by the LLM (Yashwanth, 2025). IVFFLAT (Emanuilov, 2024) uses nearest neighbour searches to find the closest data points or vectors to the user query. IVFFLAT divides vectors into clusters using a K-means algorithm, shown in Figure 19 (Emanuilov, 2024). IVFFLAT was selected for this research due to its low resource requirements and build time (Rjzv, 2023). Figure 20 shows the SQL code for indexing.

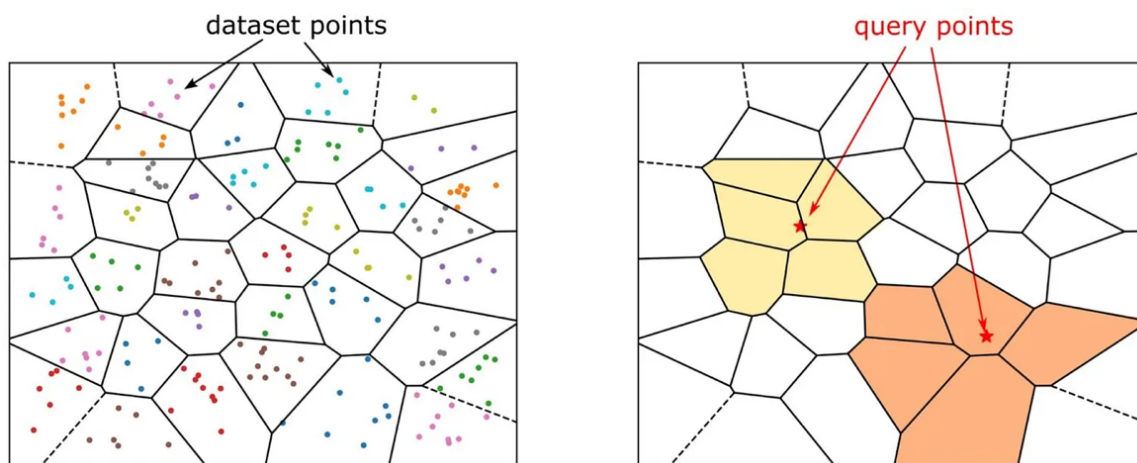


Figure 19: IVFFLAT Indexing Structure (Emanuilov, 2024)

SQL Indexing PostgreSQL Code

```
CREATE INDEX ON millie.chatbot
USING ivfflat (place_name_vector) WITH (lists = 100);

CREATE INDEX ON millie.chatbot
USING ivfflat (latitude_vector) WITH (lists = 100);

CREATE INDEX ON millie.chatbot
USING ivfflat (longitude_vector) WITH (lists = 100);

CREATE INDEX ON millie.chatbot
USING ivfflat (description_vector) WITH (lists = 100);

CREATE INDEX ON millie.chatbot
USING ivfflat (link_vector) WITH (lists = 100);
```

Figure 20: Indexing SQL GeoBot Code

IVFFLAT Limitations and an Alternative Approach

However, utilising IVFFLAT is undoubtedly a significant limitation of GeoBot when compared to the superiority of other Indexing methods. HNSW navigates through graph levels to find the closest points to the query (Pinecone, n.d), shown in Figure 21. HNSW is one of the best performing vector indexing algorithms and performs strongest at query performance compared to IVFFLAT (Gupta, 2023). If this research was repeated HNSW would be selected to improve GeoBot and enable better LLM retrieval.

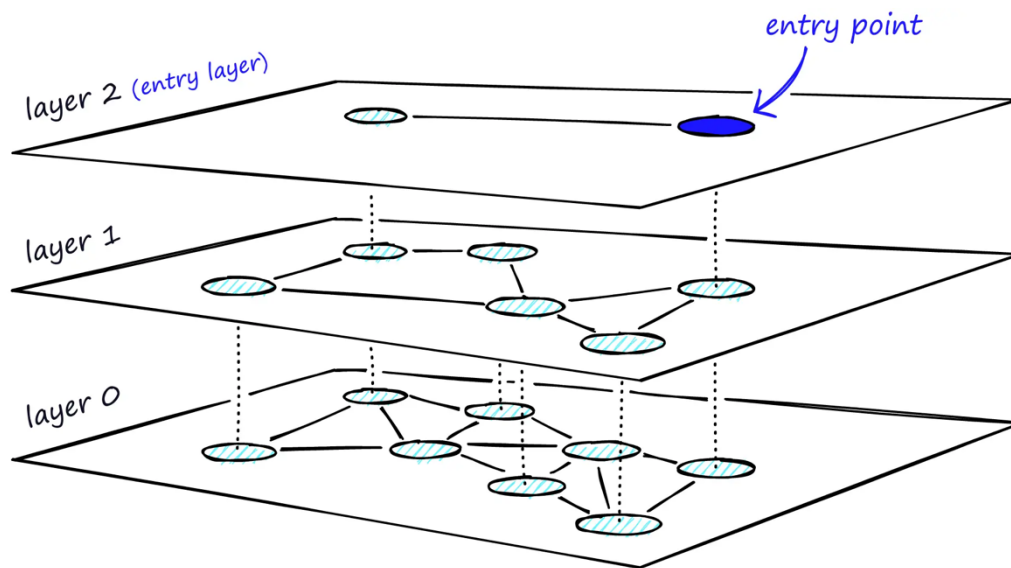


Figure 21: HNSW (Pinecone, n.d) Indexing Structure

Final Processed PostgreSQL Table Structure

Table 17 showcases the final PostgreSQL database structure for GeoBot after Chunking, Embedding and Indexing. This process enhanced the data retrieval efficiency and contextual accuracy of the LLM data retrieval, improving GeoBot responses.

Table 17: GeoBot PostgreSQL Table Structure after Chunking, Embedding and Indexing

Column	Type	Null	Default	Example
Row_ID	Integer	Not Null	Nextval ('millie.chatbot_row_id_seq'::regclass)	1
Place_Name	Text			Edinburgh
Latitude	Text			55.9523
Longitude	Text			-3.1882
Description	Text			Built on a group of hills, it is situated between the Firth of Forth...
Link	Text			https://www.scottish-places.info/towns/townfirst337.html
Place_Name_Vector	Vector(1536).		The vectorised columns contain a 1536-dimensional vector (OpenAI model default), representing the semantic embedding of the row information.	[0.15, -0.23, 0.78, 0.04, -0.56]
Latitude_Vector	Vector(1536)			[0.18, -0.10, 0.78, 0.44, -0.56]
Longitude_Vector	Vector(1536)			[0.35, -0.21, 0.70, 0.54, -0.51]
Description_Vector	Vector(1536)			[0.92, -0.13, 0.79, 0.90, -0.46]
Link_Vector	Vector(1536)			[0.19, -0.23, 0.48, 0.84, -0.06]

Flask App Methodology

Once the data has been processed in the PostgreSQL database, it must be retrieved and utilised by the OpenAI LLM so that a satisfactory response can be generated. Flask is a light web framework for Python, a software platform that enables web applications to be built (Fox, 2023). The Flask App methodology (Appendix C) was chosen to act as a bridge between the user, the PostgreSQL database and the LLM to ensure accurate responses were created. Without GeoBot's Flask app, users wouldn't have a web interface to interact with.

Flask was chosen over alternatives such as Django and FastAPI (Trujillo, 2024) due to its simplicity and easy integration with Python (Fox, 2023). Figure 22 shows GeoBot's Flask App workflow, from the user accessing the website to the OpenAI LLM generating a response. The two main components of the Flask App Backend elements will be discussed: the processing of the user query (similarity searching) and response generation (results). An additional feature conversation history will also be discussed.

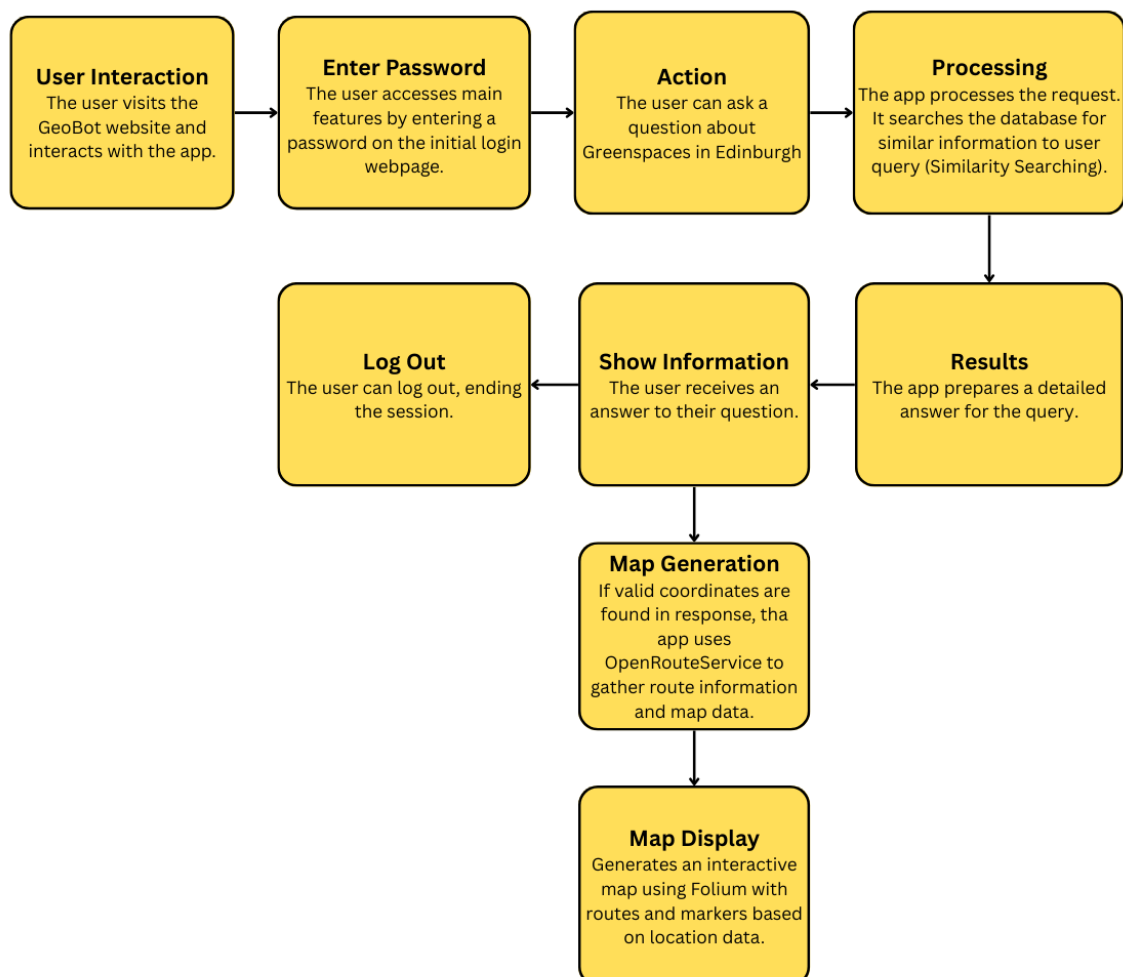


Figure 22: GeoBot Flask App Workflow

Processing: Similarity Searching

Similarity Search finds the most similar vectors to a query vector in a database (HopsWorks, 2023). This was chosen to accurately identify and retrieve the most relevant geographical data from the database, achieving high GeoBot response relevance. Euclidean Distance was chosen (code in figure 23) as it can be used to accurately calculate the distance between coordinates (Chugani, 2024). It measures the straight-line distance between two vectors (Mahmood, 2024). However, an alternative, Cosine Similarity, which measures the cosine of the angle between two vectors (Prakash, 2023), would have been the strongest similarity search option. Cosine Similarity does not consider vector magnitude or size when comparing vectors of different lengths, making it more suitable for GeoBot as the PostgreSQL database information can vary in size. If this research was repeated Cosine Similarity would be utilised.

Euclidean Distance

```
def vector_search(query_vector):
    # Creates database connection
    conn = get_connection()
    try:
        cursor = conn.cursor()
        #Turns query vector to list to ensure compatibility
        query_vector = list(query_vector)
        #SQL query to find places with vectors close to the query vector
        select_query = """
        # selects relevant columns
        SELECT place_name, description, latitude, longitude, link,
        # searches using euclidean distance
        LEAST(
            description_vector <-> %s::vector,
            place_name_vector <-> %s::vector,
            latitude_vector <-> %s::vector,
            longitude_vector <-> %s::vector,
            link_vector <-> %s::vector
        ) AS distance
        # From postgresQL database
        FROM millie.chatbot
        # Sorts results by the distance from the query vector.
        ORDER BY distance
        # Selects only the top 10 results for increased processing speed.
        LIMIT 10;
        """
        #Executes the query
        cursor.execute(select_query, (query_vector,) * 5)
        results = cursor.fetchall()
        #Closes database connection
        cursor.close()
    return results
```

Figure 23: GeoBot Euclidean Distance code excerpt (Appendix C)

Results: GeoBot Response Generation

Once the relevant database vectors have been processed and retrieved by the similarity search, OpenAI's LLM GPT-4 produces a response. The retrieved GeoBot RAG database information and user query are sent to GPT-4, which processes this data to generate a response that is then displayed to the user through the website. To achieve a geographically relevant and consistent response, GPT-4 must be prompted. A prompt is a set of instructions given to the LLM to specify the response content, structure and tone (Sheldon and Yasar, 2024). Role Prompting was chosen to achieve an increase in the quality and accuracy of responses; Kong *et al* (2023) evidence that the accuracy of their OpenAI ChatGPT responses increased from 53.5% to 63.8% when utilising role prompting. Figure 24 shows the main components required for effective role prompting, with Figure 25 showing those criteria implemented into GeoBot.

Role Prompting was essential for GeoBot as it ensured the response always contained coordinate information for subsequent map generation, as well as relevant Gazetteer links so the user can access further information and distance calculations. However, this could be further improved by providing ideal response examples and a more detailed persona role by including specific information about the intended user.

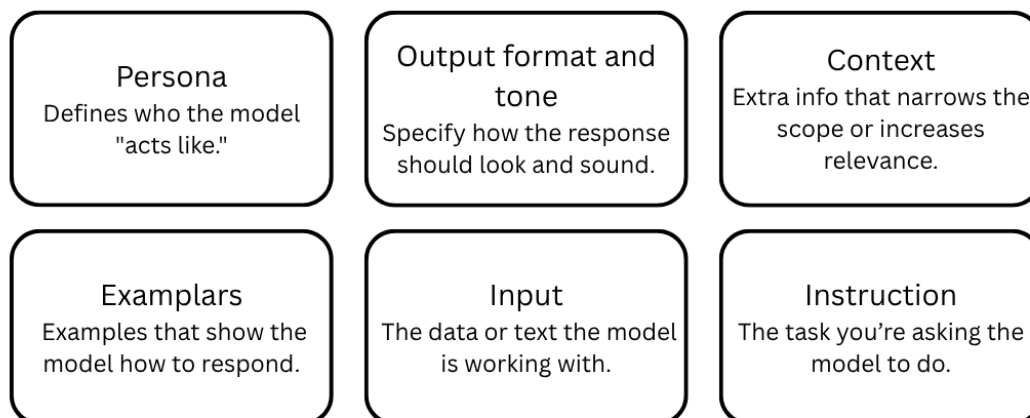


Figure 24: Role Prompting Criteria (Information from Cheung, 2024)

```

{
  "role": "system",
  "content": (
    # Persona: Defines who the model "acts like"
    "You are a geographical Retrieval augmented generation (RAG) chatbot and formatting assistant."

    # Instruction: The task you're asking the model to do
    " Answer any question, but you know about Edinburgh Greenspaces, any other knowledge is from your base OpenAI knowledge and not from M.I.L.L.I.E database information.\n\n"

    # Output Format & Tone: How the response should look and sound
    "Always format the information as follows, using UK English Spelling:\n\n"

    # Exemplars: Examples that show the model how to respond
    "1. Provide a detailed description. The response should be well-structured, clear, and professional.\n\n"

    # Output Format & Exemplars: Specific format examples for place-related info
    "2. In every response include and list the coordinates and link for every place name in this specific format:"
    "   - Coordinates for [Place Name]: Latitude: [Latitude], Longitude: -[Longitude]. Follow this [link] for more information about Place Name. Only attach Gazetteer for Scotland Links.\n\n"

    # Instruction: Distance calculation method
    "3. If providing a distance calculation, use this simplified formula for straight-line distances:\n"
    "   
$$d = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

    "   Convert this to miles. Do not explain the methodology."
    "   Note: The chatbot's map function provides distance calculations that consider local infrastructure.\n\n"
  )
},
{
  "role": "user",
  # Input: The data or text the model is working with
  "content": f"Context: {context}\nQuestion: {input_text}"
}

```

Figure 25: GeoBot Role Prompting Code (Appendix C)

Conversation History

To increase GeoBot's quality, Conversation History was also implemented as it is essential to ensure a more informed GeoBot response (Pinecone, n.d). In this research, Conversation History is utilised by storing each interaction in a session variable within the Flask App. This variable, session ['conversation_history'], accumulates chatbot responses and user inputs, preserving them for the duration of the session to maintain the conversation context (Appendix C). Figure 26 shows the successful implementation of the conversation history.

GeoBot: Welcome to GeoBot! I know about Edinburgh Greenspaces and use Gazetteer for Scotland information to inform my responses. How can I assist you today?

User: Where is Edinburgh?

GeoBot:

Edinburgh is the capital city of Scotland, located in the southeastern part of the country. It is situated on the southern shore of the Firth of Forth, facing the North Sea. Well-known for its historic and architectural landmarks, Edinburgh is a major cultural and commercial hub.

Coordinates for Edinburgh: Latitude: 55.9523, Longitude: -3.1882. Follow this [link](#) for more information about Edinburgh.

User: What did I just ask you?

GeoBot: You asked about the location of Edinburgh.

Figure 26: GeoBot Conversation History Showcase

Map Methodology

Map Feature Justification

As part of GeoBot's Flask App (Appendix C), there is an option for a map to be created once a response has been generated. Created in the Backend infrastructure, this is then displayed on the GeoBot Frontend website interface. This feature was included to provide additional context on the spatial relationships of the specified greenspaces in GeoBot's responses, achieving a richer response. Esri (n.d) state that maps allow users to 'explore, visualise spatial data and see trends over space and time', all aspects which can enhance the geographical concepts instilled in GeoBot's text responses.

Map Creation Methodology

Figure 27 shows the chosen methodology for the map creation. This uses Folium (n.d), a Python library that presents data manipulated through Python on an interactive leaflet map. Leaflet is an open-source JavaScript library for interactive maps (Nwachukwu, 2023). Leaflet was chosen due to its simplicity and useability, as well as being free to use.

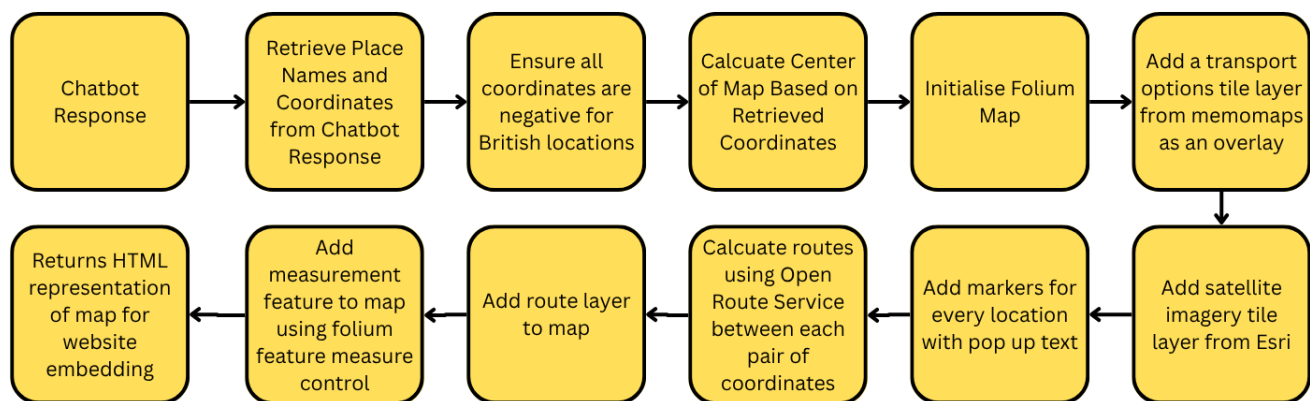


Figure 27: Overall Map Creation Methodology (Code In Appendix C)

Firstly, the relevant map data was extracted from the GeoBot response through the 're' module in Python, which identifies regular expression operations. This included place names and coordinates for the place markers, as well as hyperlinks. Once the coordinates have been extracted and the markers created the routes between the markers were calculated using Open Route Service. Open Route Service (n.d) uses data from Open Street Map, a free and editable map of the world, to provide distance calculations. Open Route Service was chosen as it is free to use and has no usage limits (Krishna, 2023). Route calculations, a measure tool, transport options tile layer and pop-up markers were all additions to increase the interactivity of the map.

Map Creation Results

Figure 28 and 29 evidence the maps created for the query ‘Where are some notable Edinburgh Greenspaces?’. This map creation methodology has achieved maps which are highly interactive and increase user engagement, with all respondents of the GeoBot user survey stating that the generated maps were easy to navigate and always accurate (Appendix H). The dynamic routing with Open Route Service is especially effective, with the route altering depending on the transport option (walking, car and cycling).

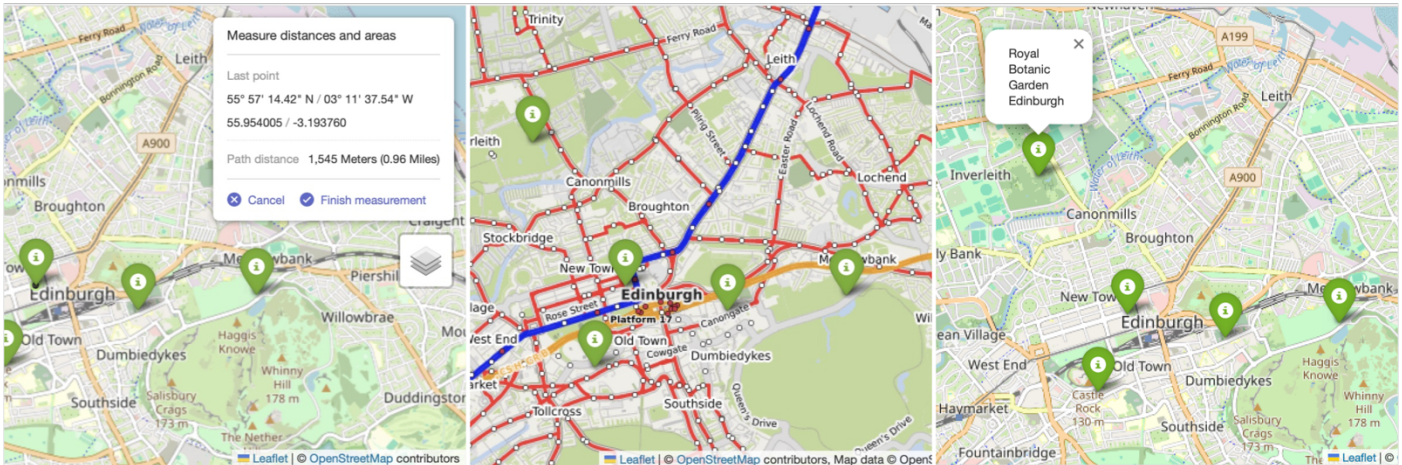


Figure 28: Measurement Tool (Left), Transport Options Tile Layer (Centre) And Pop-Up Markers (Right)

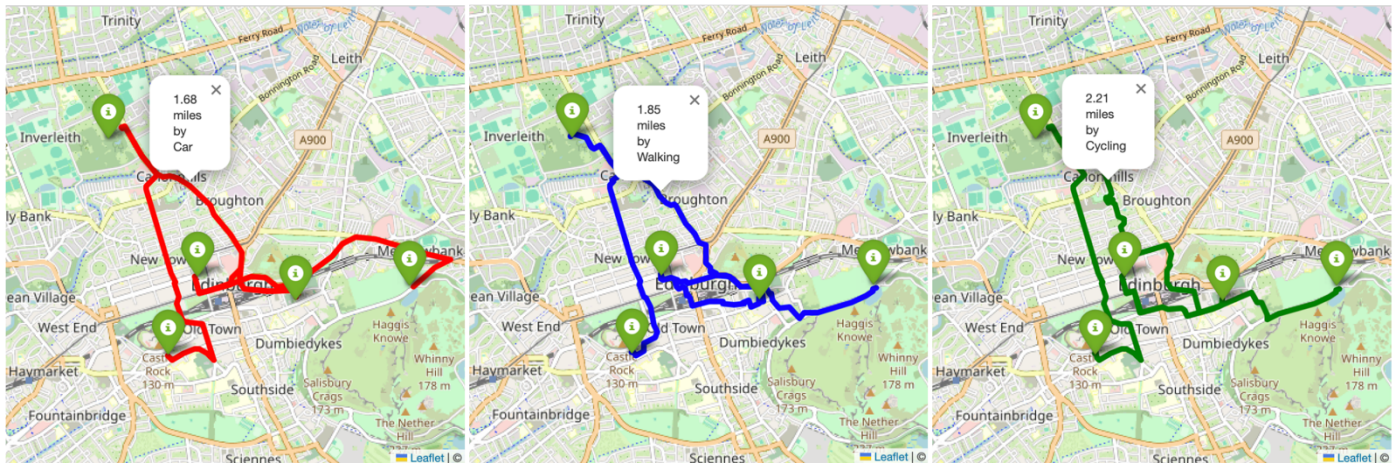


Figure 29: Route Calculations By Car (Left), Walking (Centre) And Cycling (Right)

Map Feature Improvements

Despite these maps undoubtedly providing a strong user experience, there are still improvements that could be made. Currently the greenspaces are shown through a green marker which is not precise and can blend into the background. To remedy this, a polygon showing an outline of the greenspace, made from a geojson file, would provide additional context for the user.

Additionally, the addition of additional map tiles would be beneficial. Mapbox (n.d.a) allows developers to create custom maps using the JavaScript library GL JS and even 3D environments, with can be added as tile layers to the leaflet map, shown in Figure 30. This system has been used to support BMW in car navigation and optimised travel routing for Kraken engineers, increasing efficiency by 88% (Mapbox, n.d.b). However, Mapbox is closed source and offers a pay as you go model, which costs \$0.30 per user if the monthly users exceed 100 (Mapbox, n.d.c). This rapidly decreases the scalability of GeoBot and could incur significant costs, however this sophisticated technology would be preferable for a commercial project.

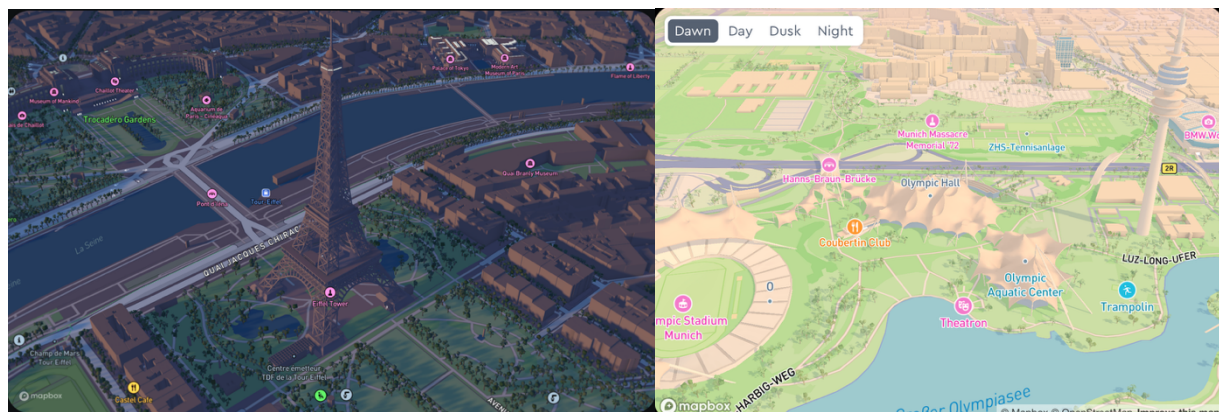


Figure 30: Mapbox 3d Model (Left) And Digital Mapping (Right) Showcase (Mapbox, n.d.b)

Further alternate datasets could also be utilised to improve the map feature. The Scottish Government (n.d) holds a wealth of open-source spatial data on subjects including the environment, census data and transport. In particular, the shapefiles for Edinburgh conversation areas and greenbelt boundaries would perfectly complement the existing Gazetteer greenspace database. To do this, these shapefiles would be converted to GeoJson files using the GeoPanda function, so that Folium could read and import the shapefile.

Alternative Map Creation Methodologies

Google Maps provides an API Key which enables extensive features, such as directions, street view and real-time traffic data (Google Maps, n.d), more so than the chosen Leaflet map. However, this was not chosen because as it is closed source the pricing can substantially fluctuate depending on usage and resource requirements.

As an alternative, a static map methodology using Cartopy was also explored. Cartopy (Scitools, n.d) is a python package designed for geospatial processing to produce maps. Figure 31 shows the tested cartopy map code. This approach was ultimately discarded as cartopy only outputs static maps. For an interactive website such as GeoBot this would not be suitable as it would reduce user interaction.

```

def plot_map(coordinates, extent):

    # making sure it is in the British Coordinate System
    bng = ccrs.OSGB()

    # Using OSM Base Layer
    map_image = OSM()

    # setting figure size
    fig = plt.figure(figsize=(10, 15))

    ax = fig.add_subplot(1, 1, 1, projection=bng)
    ax.set_extent(extent, crs=ccrs.PlateCarree())
    ax.add_image(map_image, 8, alpha=0.6)

    # setting map colour
    map_colour = plt.cm.get_cmap('tab20', len(coordinates))
    legend_names = []

    # for every location and associated latitude and longitude set a colour
    for index, (location, (latitudes, longitudes)) in enumerate(coordinates.items()):
        colour = map_colour(index)

        # setting marker size and plotting markers
        for latitude, longitude in zip(latitudes, longitudes):
            ax.plot(longitude, latitude, 'o', transform=ccrs.PlateCarree(), color=colour,
markersize=10)

        # setting legend names based on location provided
        legend_names.append(Patch(facecolor=colour, edgecolor='black', label=f'{index + 1}'
= {location.strip()}'))

    # formatting legend
    ax.legend(handles=legend_names, title='Areas', loc='lower center', handleheight=1,
framealpha=0.8,
        fontsize='xx-small', ncol=1, bbox_to_anchor=(0.5, -0.1))

    # setting title
    ax.set_title('Map')

    # plotting map
    plt.show()

```

Figure 31: Alternative Methodology Cartopy Map Code Extract

Chapter 5: GeoBot's Frontend Architecture

Frontend Architecture Overview

GeoBot's Frontend architecture, the part of a website that users directly interact with, consists of two webpages: a login page (Appendix D) where the user enters a password to prevent unauthorised access and the main webpage (Appendix E) which consists of two sections: a conversation section where the user can interact with GeoBot and a map generation section, allowing users to enter queries and view responses seamlessly. Including a Frontend for GeoBot is essential because it provides users with an intuitive and accessible interface. Figure 32 shows the main GeoBot interface, while Figure 33 shows the login page.

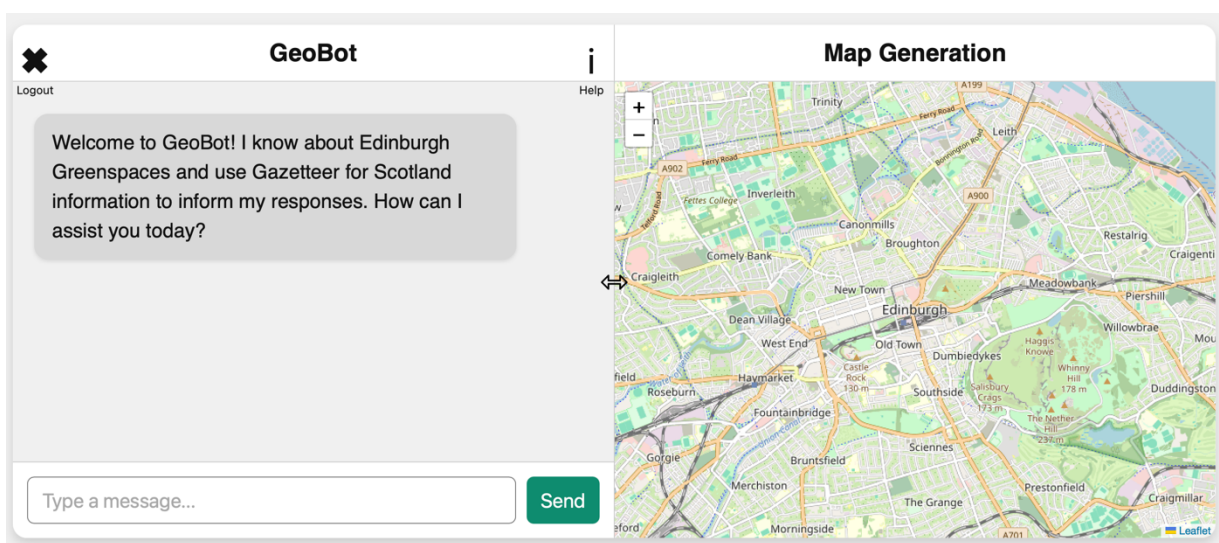


Figure 32: GeoBot Main Website Interface

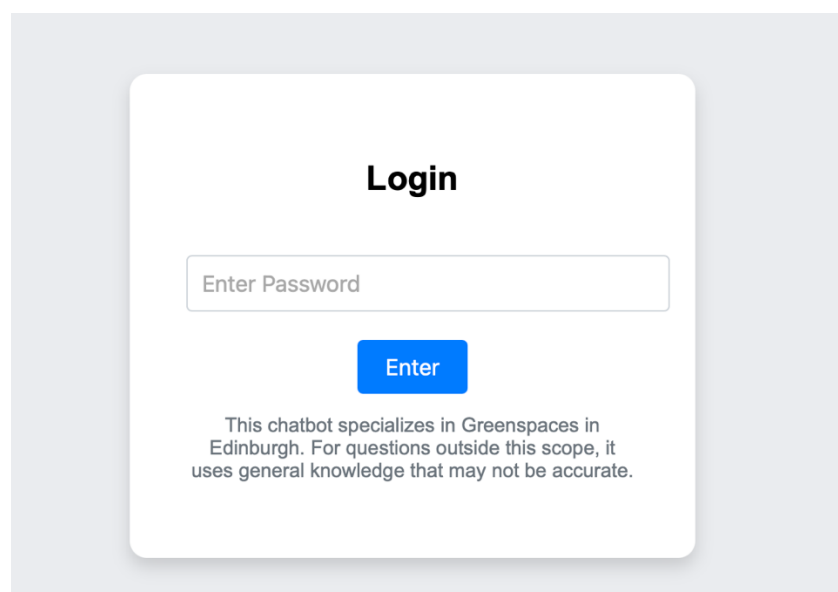


Figure 33: GeoBot Login Page Interface

Interface Design Methodology and Justification

A strongly designed interface is essential as can have a detrimental, or positive, effect on user engagement and experience (Sketch, 2022). GeoBot's interface was grounded in the principles of UI design. User Interface (UI) design is the process of constructing interfaces, focusing on stylistic elements and the user experience (Interaction Design Foundation, 2016), the key elements of which are shown in Figure 34.



Figure 34: Chatbot UI Design (Information From IBM, 2023)

GeoBot's design was mainly inspired by ELM and OpenAI's GPT-4. The aim was to create an accessible and minimal interface that would be easy to navigate. This was achieved as GeoBot's interface draws from OpenAI's minimal modern stylistics, with an unobtrusive white background and rounded chat box corners. ELM's clear and accessible interface with large green buttons also informed GeoBot's interface. This was achieved through creating a HTML webpage, using CSS (which controls the overall visual appearance of the web page) to implement the stylistic elements.

The most challenging aspect of this interface creation was the integration of the map feature while preserving the minimal and modern stylistics. This was achieved by inputting a map generation button to the right of the send message button. Initially, this map would open in another web tab. However, this drastically reduced user understanding of the map feature as it was not visible in the chat window. The map feature was then integrated by dividing the chat window in half, so the map would always be accessible next the chat window.

Interface Limitations and Improvements

Despite this interface working successfully, there are still drawbacks with this methodology. Currently, the map generation box is always visible, decreasing the chatbot window size. This means that the GeoBot conversation often becomes compressed, especially on smaller computers. An improvement would be to make the map window only appear when the map generation button is pressed, however this was too complex

to implement for this iteration of GeoBot due to the high number of changes that would be needed to be made to the CSS styling.

Consequently, another significant limitation is that the GeoBot website is currently only useable on a large device, such as a computer. This could again be remedied by improving the design with more advanced CSS styling.

Alternative Interface Methodology

Instead of creating a HTML webpage an already created user interface could have been used. LibreChat (2025), an open-source interface, can connect with OpenAI (through providing an API Key) and provides a customisable and modern chatbot interface. Despite this being a viable option for a geographical chatbot, LibreChat was not chosen due to GeoBot requiring a map generation feature which is not supported by Librechat.

Chapter 6: GeoBot Security Considerations

Securing GeoBot was essential to achieve protection against malicious attacks such as malware and SQL Injection (Cloudflare, 2023). If successful, these attacks can steal sensitive data, eroding users trust in GeoBot. This was achieved by implementing the security considerations shown in Table 18, such as environment variables to secure the OpenAI and Open Route Service API Keys, as well as the PostgreSQL database credentials. Parametrised Queries were utilised to prevent SQL Injection (commonly used attack to steal sensitive data, such as usernames and passwords (UC Berkeley, n.d)) and Session Management was also implemented so that the session data cannot be read by unauthorised parties. Despite further improvements such as storing all passwords as Environment Variables, these combined security measures do help protect sensitive GeoBot data from malicious attacks.

Table 18: Security Consideration Methodology and Justification

Security Consideration	Description	Justification and Improvements
Environment Variables	Environment variables are user credentials set in an operating system, rather than within an application (OpenAI, n.d), meaning that the GeoBot variables cannot be maliciously accessed.	This methodology was chosen for its simplicity and lack of additional costing. However, to improve this a secrets management tool such as AWS Secrets Manager or Azure Key Vault, which are systems designed to securely store API keys, could have been utilised to provide an additional layer of protection.
Parameterised Queries	Parameterised queries (UC Berkeley, n.d) ensured that the user inputs were treated as data, not as SQL queries, effectively reducing the risk of malicious access to GeoBot.	To prevent the stealing of sensitive data from the PostgreSQL database, parameterized queries with psycopg2, a PostgreSQL adapter, were used. To further improve this, the user inputs could also be validated, such as by checking they fit an expected length or data types, instead of just

		ensuring they are treated as data.
Session Management	A secure random token was created for each session, with each session encrypted and managed by cookies in the web browser (Preis, 2024).	This was chosen so GeoBot's session data cannot be read by unauthorised parties, protecting sensitive GeoBot user data being stolen. To improve this, a session timeout could be added, however this was not implemented to enable the user to ask GeoBot long queries without a time restriction.
Login Page	A HTML Website where the user has to enter a password before accessing the main GeoBot website.	This was chosen to prevent a casual user accessing the website, not to further prevent malicious attacks. However, to improve this the password should also be stored as an environment variable instead of being hard coded in the Flask app.

Chapter 7: Results Methodology

Evaluating GeoBot, as well as the comparative No-Code chatbot, was crucial to prove that RAG Models can improve Scottish geographical responses. The No-Code chatbot was created using the Botsonic (n.d) platform that uses external data from Gazetteer for Scotland URL links to provide responses through a RAG approach. This evaluation helped identify GeoBot's strengths and weaknesses, comparing this with the no-code RAG chatbot Botsonic, informing further development and improvement (McGrath and Jonker, 2024). The full results to the test chatbot questions can be found in Appendix I.

This chapter evaluates the metrics used to evaluate GeoBot's, as well as the No-Code chatbot, performance in detail, including Semantic Similarity, Response Accuracy, Clarity, Consistency, User Satisfaction, and Task Failure Rate. The inclusion of each metric is justified with alternatives considered. Additionally, the strengths and limitations of this results methodology are then discussed and limitations and future improvements presented.

GeoBot Evaluation General Overview and Justification

Zhang (2024) states that a RAG Model should be evaluated using the criteria presented in Table 19. These metrics were chosen to provide a well-rounded evaluation.

Table 19: RAG Evaluation Metrics (Zhang, 2024) and Justification for Inclusion

Evaluation Metric	Inclusion in GeoBot Evaluation Justification
Response Relevance, Accuracy and Clarity	Ensures that the GeoBot's responses are accurate, clear, and well-structured, increasing user satisfaction.
Response Stability (Consistency)	Evaluates GeoBot's ability to provide consistent answers for the same query, ensuring response reliability.
Task Success/Fail Rate and Hallucination Rate	Measures GeoBot's effectiveness in answering queries accurately while reducing false information (hallucinations), vital for GeoBot's trustworthiness.

ChatBot Response Retrieval and Storage

These metrics were calculated using Python scripts (Appendices F and G). The two created RAG Models were tested: GeoBot and the comparative No-Code chatbot Botsonic. The base GPT-4 was also assessed as a baseline comparison to GeoBot. Their answers to the six test questions were recorded. These responses were then separated into their respective chatbot sections (GeoBot, Botsonic and base GPT-4) alongside the PostgreSQL database information corresponding to the query. For example, for the test query 'How far is it between Arthur's Seat and the Meadows in Edinburgh? Does this distance vary depending on the mode of transport used?' the responses of the three chatbots were stored, alongside the relevant database information.

Response Relevance, Accuracy and Clarity Evaluation Overview (Appendix F)

For Response Relevancy and Accuracy, three methodologies were chosen: Cosine Similarity for Response Relevance, ROUGE score for Response Accuracy and Flesch Reading Score for Response Clarity. Testing these ensured that both RAG Model responses are accurate and clear to users.

Response Relevance (Semantic Similarity)

Semantic Similarity measures the similarity between the generated response and the ground truth (database source information) (RAGAS, n.d) and was chosen to ensure that the RAG Models responses were successfully drawing upon the Gazetteer for Scotland Information.

Cosine Similarity Justification

Cosine Similarity was chosen specifically as it can compare texts of different lengths (such as the database information and the GeoBot response) by analysing the similarity between the orientation of two vectors (numeric representations of words) (Karabiber, n.d). A higher score equals greater response similarity.

Cosine Similarity Methodology

The equation for cosine similarity is shown in Figure 35.

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

Figure 35: Cosine Similarity Equation (Varun, 2020)

Firstly, the response text of the three chatbots is cleaned by removing specific keywords, punctuation, and stop words, ensuring that there are not false similarity positives. All the text is then converted to vectors, with the Cosine Similarity identifying common words between each response and the reference database information (Figure 36).

```

def cosine_similarity(vec1, vec2):
    # Finds common words between two vectors
    intersection = set(vec1.keys()) & set(vec2.keys())

    # Calculates the dot product of common words
    numerator = sum([vec1[x] * vec2[x] for x in intersection])

    # Calculates the magnitudes of both vectors
    sum1 = sum([vec1[x]**2 for x in vec1])
    sum2 = sum([vec2[x]**2 for x in vec2])

    # Calculates the denominator
    denominator = math.sqrt(sum1) * math.sqrt(sum2)

    # Return cosine similarity
    return float(numerator) / denominator if denominator else 0.0

```

Figure 36: Cosine Similarity GeoBot Test Code (Appendix F)

Alternative Response Relevance Methodology

Euclidean distance, which measures the straight-line distance between vectors (Crossman-Smith, 2025), was also considered. However, this was not chosen as it does not consider the magnitude of the vectors, which means that it may not process the diverse GeoBot database effectively.

Response Accuracy

Response Accuracy measures the accuracy of responses to the database information, instead of just how similar the response is to the Gazetteer for Scotland information. This metric was essential to evaluate the extent to which the two RAG Models produce incorrect information. If the response accuracy scores are low, then GeoBot has significantly hallucinated and has not succeeded in retrieving the correct database information.

ROUGE Score Justification

ROUGE Score evaluates the recall and precision of the response compared to the reference text (Herroros, Veasay and Papaoikonomou, 2023). This was chosen as it focused on content rather than exact matches, which is preferable for AI generated text which is not directly quoted from the database.

ROUGE Score Methodology

The ROUGE score (Santhosh, 2023) was evaluated using the Python library `rouge_score` (Figure 37). The ROUGE score consists of ROUGE-N, ROUGE-L and ROUGE-S, with the balance of these results shown as the F1 Score. ROUGE-N compares overlapping words between the comparative texts and ROUGE-L analyses the similarity of sentence structures between the texts. Additionally, ROUGE-S checks for matching words that may not be next to each other in the text (Santhosh, 2023). However, while all these were calculated, only the F1 score was considered for further analysis to ensure that the results were not too complex for reader understanding.

ROUGE Score

```
def calculate_rouge(text1, text2):  
    # Calcuates the ROUGE Score using the rouge_scorer library  
    scorer = rouge_scorer.RougeScorer(['rouge1', 'rouge2', 'rougeL'], use_stemmer=True)  
    # Calcuates the ROUGE scores between each chatbot response and the database text  
    scores = scorer.score(text1, text2)  
    return scores
```

Figure 37: ROUGE Score Test Code (Appendix F)

Alternative Response Accuracy Methodology

BLEU score, which calculates precision through the matching sequences of words (Santhosh, 2023) was also considered. However, this was ultimately discarded as it only considers exact matches, which is not suitable as AI often changes the syntax for its responses even if the retrieved database information is the same.

Response Clarity

Response Clarity evaluates the readability of the two RAG Models response. This was chosen as it was essential to achieve a high user understanding, without which the RAG Models would not provide clear informative responses.

Flesch Reading Ease Test Justification

The Flesch Reading Ease Test scores text on a 100-point scale, with the higher the score meaning it is easier to understand the document (Microsoft, n.d). This methodology was chosen due to its simplicity and its widespread use across multiple sectors including research communicators and policy writers (Readable, n.d).

Flesch Reading Ease Test Methodology

The reading score was calculated using the Python import textstat (Figure 38) (Appendix F). The average reading score for each chatbot (GeoBot, Botsonic and ELM) was then calculated by dividing the total score by the number of texts in each chatbot response batch. An average result was chosen to prevent unnecessary overly complex results; a summary for each chatbot still provides a comprehensive picture of response readability.

Flesch Reading Ease Test

```
def Calculate_Flesch_Reading_Score(text_pairs, batch_size=6):
    # For every over text pair in the three batches (GeoBot, Botsonic and ELM)
    for batch_start in range(0, len(text_pairs), batch_size):
        batch = text_pairs[batch_start:batch_start + batch_size]
        total_score = 0
        count = 0

        print(f"\nBatch {batch_start // batch_size + 1}:")
        # Calculates reading score for each text in the batch
        for i, (text1, _) in enumerate(batch, start=batch_start):
            reading_score = textstat.flesch_reading_ease(text1)
            total_score += reading_score
            count += 1

        # Calculates the average reading score for each batch
        average_score = total_score / count if count else 0
```

Figure 38: Flesch Reading Ease Test Code (Appendix F)

Alternative Response Clarity Methodology

As an alternative to the Flesch Reading Ease Test, the Flesch-Kincaid Grade Level score could have been utilised. This test rates text using the U.S. school grade system. For example, a 'score of 7.0 means that a seventh grader can understand the document' (Microsoft, n.d). However, this was not chosen as it is not suitable for a British readership as it uses the American school system, reducing result understanding.

Response Stability (Consistency)

Testing Response Stability was essential as it evaluates the extent to which the RAG Models provided consistent answers for the same query, without which there is a substantial risk of incorrect responses due to the high response variation (hallucination).

Jaccard Similarity Justification and Alternative Methodologies

Jaccard Similarity calculates the similarity between sets of text through the word intersection ratio (Jadeja, 2022). Cosine Similarity and Euclidean Distance both could have been utilised instead of Jaccard Similarity, however Jaccard Similarity was chosen due to its scale invariance (it is not affected by the size of the text datasets) and versatility (it can be applied to text data instead of purely numeric data) (Zhou, 2023).

Jaccard Similarity Methodology

Jaccard Similarity was calculated through a Python script (Appendix G). For each chatbot (GeoBot, base GPT-4 and Botsonic) the Jaccard Similarity was processed separately. Three responses for each of the six test questions were recorded for each chatbot. These responses were then paired (A & B, B & C, A & C) and the Jaccard Similarity was calculated for each pair. The three separate scores were then averaged to produce an overall score for readability purposes.

Task Success/Fail Rate and Hallucination Rate

Evaluating the Response Fail and Hallucination Rates was crucial as it measures how often the two RAG Models consistently produce accurate and informative responses.

User Feedback Survey Justification and Methodology

A User Feedback Survey (Appendix H), created using Microsoft Forms (n.d) as this enables anonymous responses, was included to achieve qualitative insights into the user experience of GeoBot and the comparative No-Code chatbot Botsonic outside of a controlled test environment. The User Feedback Survey provided strong user feedback for all the evaluation criteria, the associated questions of which are shown in Figure 39. Specifically, this survey provides Task Success/Fail Rate information and Hallucination Rate questions that the other evaluation metrics lacked.

Response Relevance, Accuracy, and Clarity Questions Question 1: Did the chatbot produce a detailed and geographically comprehensive response to each query? Question 2: Was the information retrieved by the chatbot relevant to each query? Question 10: Did the alternative chatbot produce a detailed and geographically comprehensive response to each query? Question 11: Was the information retrieved by the alternative chatbot relevant to each query? Response Stability (Consistency) Questions Consistency Assessment: Questions 1, 2, 10, and 11: If responses are consistently positive, it indicates stable performance across similar queries. Task Success/Fail Rate and Hallucination Rate Questions Question 3: Did the chatbot ever fail to successfully answer a query by providing an incorrect response or no response at all? Question 4: For what question(s) did it fail and how? Question 12: Did the alternative chatbot ever fail to successfully answer a query by providing an incorrect response or no response at all? Question 13: For what question(s) did it fail and how?

Figure 39: Microsoft Forms User Feedback Survey Relevant Questions (Appendix H)

User Feedback Survey Alternatives

As an alternative, automated error responses could have been utilised to notify when GeoBot or Botsonic failed to produce a response. For example, the 'logging' Python module could store each error message in a PostgreSQL database as it is generated. However, this was not included as it lacks the detail of a User Feedback Survey and does not allow the user to expand on why the RAG Model did not produce a satisfactory response.

Overall Result Strengths, Limitations and Improvements

This results methodology has definite advantages. Combining metrics such Cosine Similarity, ROUGE Score, Flesch Reading Ease Test and Jaccard Similarity provides a thorough chatbot response evaluation, with The User Feedback Survey also capturing insights that the other quantitative metrics missed, such as the Task Success/Fail Rate.

However, only including one metric per evaluation section, such as Cosine Similarity for response relevance, means that any weaknesses in the metrics dramatically impact the overall result quality. Inclusion of another metrics for comparison would improve the robustness of the results, despite further complicating the results and slightly diminishing user understanding.

The user evaluation of the map generation was also sidelined to the User Feedback Survey (Figure 40). This means that there is not a comprehensive evaluation of the map generation, a vital part of improving LLM Scottish geographical responses. Another survey purely for map evaluation, including ideal images of GeoBot map output, would provide more well-rounded and informative user test results.

User Feedback Map Generation Questions

Question 5: Was the generated map accurate for each query?

Question 6: How clear and accessible were the maps to navigate?

Question 7: What could be improved about the mapping?

Figure 40: Microsoft Forms User Feedback Survey Map Generation Questions (Appendix H)

Chapter 8: Technical Report Conclusion

This Technical Report showcases GeoBot's successful logical architecture and how this enables the generation of accurate Scottish geographical responses. The results methodology also provides a well-rounded critical evaluation, supporting that GeoBot outperforms the no-code RAG Model Botsonic and base GPT-4.

However, there are limitations with the GeoBot methodology, mostly in the Backend Architecture. This includes an over-reliance on the Gazetteer for Scotland and the utilised Indexing method IVFFLAT. The limited useability between devices is also due to GeoBot's Frontend interface. These limitations could be mitigated for future GeoBot iterations by expanding the database and incorporating other Indexing techniques such as HNSW, as well as refining the user interface.

Beyond the scope of this research, GeoBot could also be scaled for use in the geospatial industry. For example, GeoBot could utilise the entire Gazetteer for Scotland database, rather than its current small PostgreSQL database, to significantly improve the diversity of GeoBot's responses. This would be supported by further enhancement of the data processing architecture to enable efficient retrieval, such as improving the Chunking and Embedding processing capabilities. The map generation evaluation was also sidelined to the user feedback survey; to ensure that this feature would still be beneficial to users on a widescale map user testing would be needed. GeoBot is a strong proof of concept for a geographical RAG Model and could be a beneficial tool for the geospatial industry with further improvements.

Bibliography

Amazon Web Services. (n.d). *What is RAG? - Retrieval-Augmented Generation Explained - AWS*. Amazon Web Services. Available at: <https://aws.amazon.com/what-is/retrieval-augmented-generation/>. [Accessed: 3rd March 2025].

Anthropic. (n.d). *Models overview*. Anthropic. Available at: <https://docs.anthropic.com/en/docs/about-claude/models/overview> [Accessed: 25th June 2025].

Aryani, A. (2023). 8 Types of Prompt Engineering. Medium. Available at: <https://medium.com/@amiraryani/8-types-of-prompt-engineering-5322fff77bdf> [Accessed: 23rd July 2025].

Ashutos. (2024). *How to Choose the Right Embedding Model for Your LLM Application*. Medium. Available at: <https://medium.com/@ashutoshs81127/how-to-choose-the-right-embedding-model-for-your-llm-application-7f5b257fd389> [Accessed 25th June 2025].

Barnett, S., Kurniawan, S., Thudumu, S., Brannelly, Z., Abdelrazek, M. (2024). *Seven Failure Points When Engineering a Retrieval Augmented Generation System*. ArXiv. DOI: https://doi.org/10.48550/arXiv.2401.05856open_in_new

Beatty, S. (2024). *Tiny but mighty: The Phi-3 small language models with big potential*. Microsoft. Available at: <https://news.microsoft.com/source/features/ai/the-phi-3-small-language-models-with-big-potential/> [Accessed: 20th June 2025].

Belcic, I., Stryker, C. (2025). *Model parameters*. IBM. Available at: <https://www.ibm.com/think/topics/model-parameters> [Accessed: 20th June 2025].

Ben-Yair, S. (2025). *Introducing Google AI Ultra: The best of Google AI in one subscription*. Google. Available at: <https://blog.google/products/google-one/google-ai-ultra/> [Accessed: 25th June 2025].

Bergmann, D. (2024). *Context window*. IBM. Available at: <https://www.ibm.com/think/topics/context-window> [Accessed: 20th June 2025].

Besen, S. (2024). *The Practical Limitations and Advantages of Retrieval Augmented Generation (RAG)*. Medium. Available at: <https://medium.com/data-science/the-limitations-and-advantages-of-retrieval-augmented-generation-rag-9ec9b4ae3729> [Accessed: 20th June 2025].

Bosco, A. (2024). *Transfer Learning: Feature Extraction and Fine tuning*. Medium. Available at: <https://medium.com/data-reply-it-datatech/transfer-learning-feature-extraction-and-fine-tuning-db7d82767992> [Accessed: 20th June 2025].

Botpress. (2025). *GPT-3 vs GPT-4 | What's the difference?*. Botpress. Available at: <https://botpress.com/blog/gpt-3-vs-gpt-4-whats-the-difference> [Accessed: 25th June 2025].

Botsonic. (n.d). *AI Chatbot for Website*. Botsonic. Available at: <https://botsonic.com> [Accessed: 1st July 2025].

Buntz, B. (2025). *OpenAI says GPT-4.1 sets new 90%+ standard in MMLU reasoning benchmark*. Research & Development World. Available at: <https://www.rdworldonline.com/openai-claims-gpt-4-1-sets-new-90-standard-in-mmlu-reasoning-benchmark/> [Accessed: 25th June 2025].

Chaudhary, A. (2024). *Gain The Data Advantage With Web Scraping*. Forbes. Available at: <https://www.forbes.com/councils/forbesbusinesscouncil/2024/07/23/gain-the-data-advantage-with-web-scraping/> [Accessed: 23rd July 2025].

Chawre, H. (2023). *Fine-Tuning LLMs: Overview, Methods, and Best Practices*. Turing. Available at: <https://www.turing.com/resources/finetuning-large-language-models> [Accessed: 20th June 2025].

Chugani, V. (2024). *Understanding Euclidean Distance: From Theory to Practice*. Datacamp. Available at: <https://www.datacamp.com/tutorial/euclidean-distance> [Accessed: 26th June 2025].

Cloudflare. (2022). *What is a large language model (LLM)?* Cloudflare. Available at: <https://www.cloudflare.com/en-gb/learning/ai/what-is-large-language-model/> [Accessed: 23rd July 2025].

Cloudflare. (2023). *How to prevent DDoS attacks | Methods and tools*. Cloudflare. Available at: <https://www.cloudflare.com/en-gb/learning/ddos/how-to-prevent-ddos-attacks/> [Accessed: 23rd July 2025].

Code Academy. (2025). *What is Back-End Architecture?*. Codecademy. Available at: <https://www.codecademy.com/article/what-is-back-end-architecture> [Accessed: 11th July 2025].

Cook, J. (2024). *How To Protect Your Content From ChatGPT (7 Top Tactics)*. Forbes. Available at: <https://www.forbes.com/sites/jodiecook/2024/05/30/how-to-protect-your-content-from-chatgpt-7-top-tactics/> [Accessed: 25th June 2025].

Crossman-Smith, J. (2025). *Distance Metrics in Machine Learning - Low Code for Data Science*. Medium. Available at: <https://medium.com/low-code-for-advanced-data-science/distance-metrics-in-machine-learning-without-the-maths-994392d3995b> [Accessed: 23rd July 2025].

Deepchecks. (2024). *What are LLM Parameters? Explained Simply*. Deepchecks. Available at: <https://www.deepchecks.com/glossary/llm-parameters/> [Accessed: 20th June 2025].

DiDonato, B. (2024). *Retrieval Augmented Generation (RAG) On Google Cloud*. Medium. Available at: <https://medium.com/zencore/retrieval-augmented-generation-rag-on-google-cloud-ba58707c7b57> [Accessed: 25th June 2025].

Donovan, R. (2024). *Breaking up is hard to do: Chunking in RAG applications - Stack Overflow*. Available at: <https://stackoverflow.blog/2024/12/27/breaking-up-is-hard-to-do-chunking-in-rag-applications/> [Accessed: 25th June 2025].

Du, H., Liu, S., Zheng, L., Cao, Y., Nakamura, A., Chen, L. (2024). *Privacy in fine-tuning large language models: Attacks, defenses, and future directions*. 1, pp. 1-19. Available at: DOI: <http://doi.org/10.48550/arXiv.2412.16504> [Accessed: 20th June 2025].

ElasticSearch. (n.d). *Elasticsearch: The Official Distributed Search & Analytics Engine*. ElasticSearch. Available at: <https://www.elastic.co/elasticsearch> [Accessed: 23rd July 2025].

Emanuilov, S. (2024). *IVFFlat or HNSW index for similarity search?*. Medium. Available at: <https://medium.com/@simeon.emanuilov/ivfflat-or-hnsw-index-for-similarity-search-31d181a490a0> [Accessed: 4th August 2025].

Espejel, O. (2022). *Getting Started With Embeddings*. Hugging Face. Available at: <https://huggingface.co/blog/getting-started-with-embeddings> [Accessed: 25th June 2025].

Esri. (n.d). *The power of maps*. Esri. Available at: <https://learn.arcgis.com/en/projects/the-power-of-maps/> [Accessed: 27th June 2025].

Folium. (n.d). *Folium — Folium 0.1.dev1+g57e8eae documentation*. GitHub. Available at: <https://python-visualization.github.io/folium/latest/> [Accessed: 27th June 2025].

Fortinet. (n.d). *What Is an API Key? | API Key Definition*. Fortinet. Available at: <https://www.fortinet.com/uk/resources/cyberglossary/api-key> [Accessed: 25th June 2025].

Fox, L. (2023). *Why Should You Use Flask Framework For Web Development?* Medium. Available at: <https://medium.com/@lauren-fox/why-should-you-use-Flask-framework-for-web-development-f5a7233e17a6> [Accessed: 23rd July 2025].

Gal, T. (2023). *Transformer Architecture Simplified*. Medium. Available at: <https://medium.com/@theaveragegal/transformer-architecture-simplified-3fb501d461c8> [Accessed: 20th June 2025].

Gepiti, C. (2025). *Meta AI chatbot at the center of major data leak scandal*. Durovs Code. Available at: <https://durovscode.com/meta-ai-chatbot-at-the-center-of-major-data-leak-scandal> [Accessed: 20th June 2025].

Gerken, T. (2025). *Be careful with DeepSeek, Australia says - so is it safe to use?*. BBC News. Available at: <https://www.bbc.co.uk/news/articles/cx2k7r5nrupo> [Accessed: 20th June 2025].

Gitting, B. (2025). *The Meadows from The Gazetteer for Scotland*. The Gazetteer for Scotland, Available at: <https://www.scottish-places.info/features/featurefirst990.html> [Accessed 25th June 2025].

Gittings, B. (n.d). *The Gazetteer For Scotland*. Available at: <https://www.scottish-places.info> [Accessed: 23rd July 2025].

Gittings, B. (2023). *Haymarket from The Gazetteer for Scotland*. The Gazetteer For Scotland. Available at: <https://www.scottish-places.info/features/featurefirst8564.html> [Accessed: 3rd July 2025].

Google Maps. (n.d). *Google Maps*. Google Maps. Available at: <https://www.google.com/maps> [Accessed: 23rd July 2025].

Google. (2023). *Gemini*. Google. Available at: <https://gemini.google.com/app>. [Accessed: 3rd March 2025].

Google. (2025). *Vertex AI RAG Engine overview*. Google. Available at: <https://cloud.google.com/vertex-ai/generative-ai/docs/rag-engine/rag-overview> [Accessed: 26th May 2025].

GPThub. (2022). *OpenAI Unveils a Powerful, Cost-Effective, and User-Friendly Embedding Model*. GPThub. Available at: <https://www.gpthub.ai/ai-news-and-announcements/entrepreneurship-ideas-for-young-and-ambitious-people/> [Accessed: 9th June 2025].

Gupta, E. (2023). *Cloud SQL for PostgreSQL: Faster similarity search performance with pgvector indexes*. Google. Available at: <https://cloud.google.com/blog/products/databases/faster-similarity-search-performance-with-pgvector-indexes> [Accessed: 25th June 2025].

Greyling, C. (2024). *Catastrophic Forgetting In LLMs*. Medium. Available at: <https://cobusgreyling.medium.com/catastrophic-forgetting-in-llms-bf345760e6e2> [Accessed 23rd July 2025].

Hafnar, D. (2024). *Building a Serverless RAG System with BigQuery and Gemini*. Medium. Available at: <https://medium.com/@dhafnar/building-a-serverless-rag-system-with-bigquery-and-gemini-12e66328f622> [Accessed: 25th June 2025].

- Harkar, S. (2025). *RAG Techniques*. IBM. Available at: <https://www.ibm.com/think/topics/rag-techniques> [Accessed: 20th June 2025].
- Heath, A. (2023). *Web Scraping 101: Tools, Techniques and Best Practices*. Geek Culture. Available at: <https://medium.com/geekculture/web-scraping-101-tools-techniques-and-best-practices-417e377fbeaf> [Accessed: 4th July 2025].
- Hernandez, J. (2023). *What are Vector Embeddings in AI and LLM?*. Medium. Available at: <https://devjaime.medium.com/what-are-vector-embeddings-in-ai-and-llm-d40732c27f82> [Accessed: 25th June 2025].
- Herroros, Q. Veasey, T., Papaoikonomou, T. (2023). *RAG evaluation metrics: A journey through metrics*. Search Labs. Available at: <https://www.elastic.co/search-labs/blog/evaluating-rag-metrics> [Accessed: 23rd July 2025].
- Hopsworks. (2023). *What is Similarity Search?*. Hopsworks. Available at: <https://www.hopsworks.ai/dictionary/similarity-search> [Accessed: 27th May 2025].
- Hughes, A. (2023). *Phi-2: The surprising power of small language models*. Microsoft Research. Available at: <https://www.microsoft.com/en-us/research/blog/phi-2-the-surprising-power-of-small-language-models/> [Accessed: 20th June 2025].
- IBM. (2023). *Chatbot design*. IBM. Available at: <https://www.ibm.com/think/topics/chatbot-design> [Accessed: 23rd July 2023].
- Interaction Design Foundation. (2016). *What is User Interface (UI) Design?*. Interaction Design Foundation. Available at: https://www.interaction-design.org/literature/topics/ui-design?srltid=AfmBOopXQPGSXWupJYDMGxvl9pJxz5Uw028Ns6nYn_Kx1_jBWkgZX_rn [Accessed: 1st July 2025]
- Iterations, A. (2025). *LLM Cost Estimation Guide: From Token Usage to Total Spend*. Medium. Available at: <https://medium.com/@alphaiterations/llm-cost-estimation-guide-from-token-usage-to-total-spend-fba348d62824> [Accessed: 11th July 2025].
- Jikadara, B. (2025). *Exploring the Different Types of RAG in AI*. Medium. Available at: <https://bhavikjikadara.medium.com/exploring-the-different-types-of-rag-in-ai-c118edf6d73c> [Accessed: 20th June 2025].
- Jadeja, M. (2022). *Jaccard Similarity Made Simple: A Beginner's Guide to Data Comparison*. Medium. Available at: <https://mayurdhvajsinhjadeja.medium.com/jaccard-similarity-34e2c15fb524> [Accessed: 24th July 2025].
- Karabiber, F. (n.d). *Cosine Similarity*. Learn Data Science. Available at: <https://www.learndatasci.com/glossary/cosine-similarity/> [Accessed: 23rd July 2025].

Kong, A., Zhao, S., Chen, H., Li, Q., Qin, Y., Sun, R., Zhou, X. (2023). *Better Zero-Shot Reasoning with Role-Play Prompting*. Arxiv.
DOI:<https://doi.org/10.48550/arXiv.2308.07702>.

Krishna, S. (2023). *Best Google Maps API Alternatives for Bespoke Mapping Features*. Sweans Technologies. Available at: <https://www.sweans.com/is-there-a-reliable-google-maps-api-alternative/> [Accessed: 27th June 2025].

Kutumbe, K. (2024). *Comprehensive Guide to Chunking in LLM and RAG Systems*. Medium. Available at: <https://kshitijkutumbe.medium.com/comprehensive-guide-to-chunking-in-llm-and-rag-systems-c579a11ce6e2> [Accessed: 23rd July 2025].

Kutz, J. (2025). *OpenAI Embeddings 101: A Perfect Guide For Data Engineers*. Airbyte. Available at: <https://airbyte.com/data-engineering-resources/openai-embeddings> [Accessed: 25th June 2025].

Li, L. (2025). *Embedding Model Comparison: text-embedding-ada-002 vs. text-embedding-3-large Across Different Dimensions*. Medium. Available at: <https://medium.com/@lilianli1922/embedding-model-comparison-text-embedding-ada-002-vs-a618116575a6> [Accessed: 23rd July 2025].

Librechat. (2025). *LibreChat*. LibreChat. Available at: <https://www.librechat.ai/> [Accessed: 1st July 2025].

Lubbad, M. (2023). *GPT-4 Parameters: Unlimited guide NLP's Game-Changer*. Medium. Available at: <https://mlubbad.com/the-ultimate-guide-to-gpt-4-parameters-everything-you-need-to-know-about-nlps-game-changer-109b8767855a> [Accessed: 20th June 2025].

Macdonald, L. (2024). *RAG Vs Fine Tuning: How To Choose The Right Method*. Monte Carlo. Available at: <https://www.montecarlodata.com/blog-rag-vs-fine-tuning/> [Accessed: 20th June 2025].

Mahmood, M. (2024). *What is Euclidean Distance?*. Medium. Available at: <https://medium.com/@mohamad.razzi.my/what-is-euclidean-distance-a4dab1b9f329> [Accessed: 26th June 2025].

Mapbox. (n.d.a). *Maps*. Mapbox. Available at: <https://www.mapbox.com/maps#integrate-maps> [Accessed: 27th June 2025].

Mapbox.com. (n.d.b). *Customer Showcase*. Mapbox. Available at: <https://www.mapbox.com/showcase> [Accessed: 27th June 2025].

Mapbox. (n.d.c). *Mapbox pricing*. Mapbox. Available at: <https://www.mapbox.com/pricing> [Accessed: 27th June 2025].

Martin-Short, R. (2024). *A Visual Exploration of Semantic Text Chunking - TDS Archive*. Medium. Available at: <https://medium.com/data-science/a-visual-exploration-of-semantic-text-chunking-6bb46f728e30> [Accessed: 25th June 2025].

Martinez, J. (2023). *Supervised Fine-tuning: customizing LLMs*. Medium. Available at: <https://medium.com/mantisnlp/supervised-fine-tuning-customizing-llms-a2c1edbf22c3> [Accessed: 20th June 2025].

Mastering LLMs. (2024). *11 Chunking Strategies for RAG — Simplified & Visualized*. Medium. Available at: <https://masteringllm.medium.com/11-chunking-strategies-for-rag-simplified-visualized-df0dbec8e373> [Accessed: 25th June 2025].

Mckinsey. (2024). *What is prompt engineering?* Mckinsey. Available at: <https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-prompt-engineering> [Accessed: 23rd July 2025].

Meta. (2025). *The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation*. Meta. Available at: <https://ai.meta.com/blog/llama-4-multimodal-intelligence/> [Accessed: 25th June 2025].

Meta AI. (2025). *Meta AI*. Meta. Available at: <https://ai.meta.com/meta-ai/> [Accessed: 20th June 2025].

Microsoft Azure. (n.d). *Key Vault*. Microsoft Azure. Available at: <https://azure.microsoft.com/en-us/products/key-vault> [Accessed: 1st July 2025].

Microsoft. (n.d.a). *What is a Cloud Service Provider?*. Microsoft Azure. Available at: <https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-a-cloud-provider> [Accessed: 25th June 2025].

Microsoft. (n.d.b). *Get your document's readability and level statistics*. Microsoft Support. Available at: https://support.microsoft.com/en-us/office/get-your-document-s-readability-and-level-statistics-85b4969e-e80a-4777-8dd3-f7fc3c8b3fd2#__toc342546557 [Accessed: 12th July 2025].

Mishra, A. (2024). *Five Levels of Chunking Strategies in RAG*. Medium. Available at: https://medium.com/@anuragmishra_27746/five-levels-of-chunking-strategies-in-rag-notes-from-gregs-video-7b735895694d [Accessed: 25th June 2025].

Murel, J., Kavlakoglu, E. (2024). *Reinforcement Learning*. IBM. Available at: <https://www.ibm.com/think/topics/reinforcement-learning> [Accessed: 20th June 2025].

Naminas. (2024). *LLM Architecture: Possible Model Configurations*. Labelyourdata. Available at: <https://labelyourdata.com/articles/llm-architecture> [Accessed: 20th June 2025].

National Library of Medicine. (2022). *Web Scraping*. National Library of Medicine. Available at: <https://www.nlm.gov/guides/data-glossary/web-scraping> [Accessed: 4th July 2025].

Nwachukwu, C. (2023). *Building Interactive Maps with Folium, Python*. Medium. Available at: <https://medium.com/@thechibuzornwachukwu/creating-interactive-maps-the-easy-way-with-python-folium-and-leaflet-js-f5bcc24862f> [Accessed: 27th June 2025].

Octoparse. (n.d). *Web Scraping Tool & Free Web Crawlers*. Octoparse. Available at: <https://www.octoparse.com> [Accessed: 23rd July 2025].

Open Route Service (n.d.). *Openrouteservice*. Open Route Service. Available at: <https://openrouteservice.org> [Accessed: 27th June 2025]

OpenAI. (2025). *Pricing*. OpenAI. Available at: <https://openai.com/api/pricing/> [Accessed: 20th June 2025].

OpenAI. (2023). *GPT-4*. OpenAI. Available at: <https://openai.com/index/gpt-4-research/> [Accessed: 23rd July 2025].

OpenAI. (2024). *OpenAI Platform*. OpenAI. Available at: <https://platform.openai.com/docs/bots> [Accessed: 4th July 2025].

OpenAI. (n.d). *Best Practices for API Key Safety*. OpenAI. Available at: <https://help.openai.com/en/articles/5112595-best-practices-for-api-key-safety> [Accessed: 1st July 2025].

Oracle. (2014). *What is a relational database?* Oracle. Available at: <https://www.oracle.com/uk/database/what-is-a-relational-database/> [Accessed: 23rd July 2025].

Pichai, S. and Hassabis, D. (2023). *Introducing Gemini: our largest and most capable AI model*. Google. Available at: <https://blog.google/technology/ai/google-gemini-ai/> [Accessed: 23rd July 2025].

Pinecone. (n.d). *Conversational Memory for LLMs with Langchain*. Pinecone Available at: <https://www.pinecone.io/learn/series/langchain/langchain-conversational-memory/> [Accessed: 26th June 2025].

PlayAI. (2025). *No-Code Chatbot: Build AI Chatbots Without Programming*. PlayAI. Available at: <https://play.ht/blog/no-code-chatbot/> [Accessed: 11th July 2025].

PostgreSQL. (n.d). *PostgreSQL: The world's most advanced open source database*. PostgreSQL. Available at: <https://www.postgresql.org> [Accessed: 23rd July 2025].

Prakash, A. (2023). *Understanding Cosine Similarity: A key concept in data science*. Medium. Available at: <https://medium.com/@arjunprakash027/understanding-cosine-similarity-a-key-concept-in-data-science-72a0fcc57599> [Accessed: 26th June 2025].

Preis, A. (2024). *How Session Management Works and Why It's Important*. Ping Identity. Available at: <https://www.pingidentity.com/en/resources/blog/post/session-management.html> [Accessed: 23rd July 2025].

Proudfoot, O. (2024). *What is Claude AI, and how does it compare to ChatGPT?* Pluralsight. Available at: <https://www.pluralsight.com/resources/blog/ai-and-data/what-is-claude-ai> [Accessed: 25th June 2025].

Python Coding. (2024). *Securing Your Python Programs*. Medium. Available at: <https://pythonclcoding.medium.com/securing-your-python-programs-c41ec49ae871> [Accessed: 1st July 2025].

Python. (n.d). *What Is Python? Executive Summary*. Python. Available at: <https://www.python.org/doc/essays/blurb/> [Accessed: 23rd July 2025].

RAGAS. (n.d). *Semantic Similarity*. RAGAS. Available at: https://docs.ragas.io/en/latest/concepts/metrics/available_metrics/semantic_similarity/ [Accessed: 7th June 2025].

Ray, S. (2023). *Samsung Bans ChatGPT among Employees after Sensitive Code Leak*. Forbes. Available at: <https://www.forbes.com/sites/siladityaray/2023/05/02/samsung-bans-chatgpt-and-other-chatbots-for-employees-after-sensitive-code-leak/> [Accessed: 23rd July 2025].

Readable. (n.d). *Flesch Reading Ease and the Flesch Kincaid Grade Level*. Readable. Available at: <https://readable.com/readability/flesch-reading-ease-flesch-kincaid-grade-level/> [Accessed: 23rd July 2025].

Rein, D., Hou, B.L., Stickland, A.C., Petty, J., Pang, R.Y., Dirani, J., Michael, J., Bowman, S. (2023). *GPQA: A Graduate-Level Google-Proof Q&A Benchmark*, 1, pp. 1-28. Available at: DOI: <https://doi.org/10.48550/arXiv.2311.12022> [Accessed: 25th June 2025].

Richards, D. (2024). *RAG in the Cloud: Comparing AWS, Azure, and GCP for Deploying Retrieval Augmented Generation Solutions – News from generation RAG*. Rag About It. Available at: <https://ragaboutit.com/rag-in-the-cloud-comparing-aws-azure-and-gcp-for-deploying-retrieval-augmented-generation-solutions/> [Accessed: 25th June 2025].

Rjzv. (2023). *Vector Indexes in Postgres using pgvector: IVFFlat vs HNSW*. Tembo. Available at: <https://www.tembo.io/blog/vector-indexes-in-pgvector> [Accessed 25th June 2025].

Saeed, T. (2024). *Building a RAG solution using Amazon Bedrock and Amazon OpenSearch Serverless*. Medium. Available at: https://medium.com/@tahir.saeed_46137/building-a-rag-solution-using-amazon-bedrock-and-amazon-opensearch-serverless-7e6f7f4f98dd [Accessed: 25th June 2025].

Santhosh, S. (2023). *Understanding BLEU and ROUGE score for NLP evaluation*. Medium. Available at: <https://medium.com/@sthanikamsanthosh1994/understanding-bleu-and-rouge-score-for-nlp-evaluation-1ab334ecadcb>. [Accessed: 23rd July 2025].

Schwaber-Cohen, R. (2023). *Chunking Strategies for LLM Applications*. Pinecone. Available at: <https://www.pinecone.io/learn/chunking-strategies/> [Accessed: 25th June 2025].

Scitools. (n.d). *Introduction — cartopy 0.19.0rc2.dev8+gd251b2f documentation*. Scitools. Available at: <https://scitools.org.uk/cartopy/docs/latest/> [Accessed: 27th June 2025].

Sheldon, R., Yasar, K. (2024). *What is an AI Prompt?*. TechRadar. Available at: <https://www.techtarget.com/searchenterpriseai/definition/AI-prompt> [Accessed: 26th June 2025].

Shone, O. (2025). *5 key features and benefits of retrieval augmented generation (RAG) | The Microsoft Cloud Blog*. The Microsoft Cloud Blog. Available at: <https://www.microsoft.com/en-us/microsoft-cloud/blog/2025/02/13/5-key-features-and-benefits-of-retrieval-augmented-generation-rag/> [Accessed 9th June 2025].

Sketch. (2022). *What Is UI design? and Why Is It important?*. Sketch. Available at: <https://www.sketch.com/blog/what-is-ui-design/> [Accessed: 1st July 2025].

Spheron. (2024). *How to Build an LLM from Scratch: A Step-by-Step Guide*. Spheron Network. Available at: <https://blog.spheron.network/how-to-build-an-llm-from-scratch-a-step-by-step-guide> [Accessed: 20th June 2025].

The National Records Of Scotland. (n.d). *NRS Web Continuity Service*. The National Records Of Scotland. Available at: <https://webarchive.nrscotland.gov.uk/20241128124517/https://www.nrscotland.gov.uk/files//statistics/council-area-data-sheets/city-of-edinburgh-council-profile.html> [Accessed: 3rd July 2025].

The Scotland's Environment Agency. (n.d). *Scotland's environment web*. The Scotland's Environment Agency. Available at: <https://www.environment.gov.scot> [Accessed: 23rd July 2025].

The Scottish Government. (n.d). *Useful data sources*. The Scottish Government. Available at: <https://www.environment.gov.scot/data/useful-data-sources/#peopleAndTheEnvironment> [Accessed: 3rd July 2025].

Tiedrich, L. (2024). *The AI Data Scraping challenge: How Can We Proceed Responsibly?*. OECD. Available at: <https://oecd.ai/en/work/data-scraping-responsibly> [Accessed: 4th July 2025].

Trujillo, V. (2024). *Exploring Python Web Frameworks: Alternatives to Flask*. Medium. Available at: <https://medium.com/@vanessa.trujillo/exploring-python-web-frameworks-alternatives-to-Flask-f37f1dfdc3b7> [Accessed: 11th July 2025].

TU. (2021). *Optimising pairwise Euclidean distance calculations using Python | Towards Data Science*. Towards Data Science. Available at: <https://towardsdatascience.com/optimising-pairwise-euclidean-distance-calculations-using-python-fc020112c984/> [Accessed: 23rd July 2025].

UC Berkeley. (n.d). *How to Protect against SQL Injection Attacks*. UC Berkeley. Available at: <https://security.berkeley.edu/education-awareness/how-protect-against-sql-injection-attacks> [Accessed: 1st July 2025].

Varun. (2020). *Cosine similarity: How does it measure the similarity, Maths behind and usage in Python | Towards Data Science*. Towards Data Science. Available at: <https://towardsdatascience.com/cosine-similarity-how-does-it-measure-the-similarity-maths-behind-and-usage-in-python-50ad30aad7db/> [Accessed: 23rd July 2025].

Williams, W. (2024). *OpenAI spent \$80M to \$100M training GPT-4; Chinese firm claims it trained its rival AI model for \$3 million using just 2,000 GPUs*. TechRadar. Available at: <https://www.techradar.com/pro/openai-spent-usd80m-to-usd100m-training-gpt-4-chinese-firm-claims-it-trained-its-rival-ai-model-for-usd3-million-using-just-2-000-gpus> [Accessed: 20th June 2025].

Woyera. (2024). *We tested the top 7 no-code chat bot tools. Here's what we found*. Medium. Available at: <https://medium.com/@woyera/guide-to-no-code-ai-chatbots-40d37bc6d56b> [Accessed: 27th May 2025].

Yan, B., Li, K., Xu, M., Dong, Y., Zhang, Y., Ren, Z., Cheng, X. (2025). On protecting the data privacy of Large Language Models (LLMs) and LLM agents: A literature review. *High-Confidence Computing*, 5(2), p.100300. DOI:<https://doi.org/10.1016/j.hcc.2025.100300>.

Yashwanth, S. (2025). *Retrieval Augmented Generation (RAG) 05 :- Indexing in LLM RAG*. Medium. Available at: https://medium.com/@yashwanths_29644/retrieval-augmented-generation-rag-05-indexing-strategies-4b822513ea07 [Accessed 25th June. 2025].

Zhang, S. (2024). *Evaluating LLM-based chatbots: A comprehensive guide to performance metrics*. Medium. Available at: <https://medium.com/data-science-at-microsoft/evaluating-llm-based-chatbots-a-comprehensive-guide-to-performance-metrics-9c2388556d3e> [Accessed: 3rd March 2025].

Zhou, C. (2023). *Understanding Jaccard Similarity: A Powerful Tool for Data Analysis*. Medium. Available at: <https://medium.com/@conniezhou678/understanding-jaccard-similarity-a-powerful-tool-for-data-analysis-42abaaafd782> [Accessed: 23rd July 2025].

Appendices

All programming code can be found in the M Drive of Millie Stanley-Davy (s2478921). The absolute file path to the relevant folder is: /home/s2478921/geobot. This is equivalent to a disfinal folder. Any issues please contact s2478921@ed.ac.uk .

The file structure of GeoBot is as follows:

/home/s2478921/geobot/

app4.py (Flask App)

Processing/

Web_scraping.py (Web Scraping Script)

Vectorising.py (Embedding Script)

Templates/

Index2.html (Main GeoBot HTML webpage)

Login.html (HTML Login Page)

Results/

Mainresults.py (Results Evaluation Main Script)

Jaccardsimilarity.py (Jaccard Similarity Script)

Please do not run the web_scraping.py or vectorising.py files in the processing folder as this may overwrite parts of GeoBot's database. To run the results files type 'python [file_name]' in the command line in the location /home/s2478921/geobot/results/ .

The geobot website should still be active so it should not need to be ran again. The website is: <https://www.geos.ed.ac.uk/dev/geobot/main> (Password is 'GIS2025!'). However, if this is not active then the commands to run the flask app are:

1. Navigate to the file path /home/s2478921/geobot .
2. Run 'ssh devapps'
3. Run 'source venv/geobot/bin/activate' to set up virtual environment.
4. (Optional) Run 'screen -r geochatbot' to connect to virtual screen to run GeoBot even if logged off. If screen is not activated, set up with 'screen -S geochatbot'.
5. Run command: 'SCRIPT_NAME=/dev/geobot gunicorn --bind 0.0.0.0:55423 app4:app'
If this is unsuccessful try: 'SCRIPT_NAME=/dev/geobot
/home/s2478921/venv/geobot/bin/gunicorn --bind 0.0.0.0:55423 app4:app'
6. If there is an error saying gunicorn is not downloaded, the command to download it is 'pip install gunicorn'.

Further guidance can be found here:

<https://uoe.sharepoint.com/sites/GeoSciencesIT/SitePages/python-web-apps.aspx>

GeoBot information: port number: 55423, app name: geobot and module name: app4

List of Appendices

Appendix A: Web Scraping Python Script. This collects information from the Edinburgh Gazetteer For Scotland website.

Appendix B: Embedding Python Script. This converts the GeoBot database to vectors so the information can be read by the LLM.

Appendix C: Flask App Python Script. The main section of GeoBot which handles data retrieval, response generation and map generation.

Appendix D: Login Website Script.

Appendix E: Main GeoBot Website Script.

Appendix F: Results Python Script. The main results script which runs Cosine Similarity, Rouge Score and Flesch Reading Ease Test.

Appendix G: Jaccard Similarity Results Testing Python Script.

Appendix H: Microsoft Survey. The full results showcase from the user feedback survey.

Appendix I: ChatBot Test Question Results. The full results for GeoBot, comparative no-code chatbot Botsonic and base GPT-4 from the six test questions:

Question One: What are the geographical features of greenspaces in Edinburgh?

Question Two: How far is it between Arthur's Seat and the Meadows in Edinburgh? Does this distance vary depending on the mode of transport?

Question Three: Which greenspaces in Edinburgh are connected with famous figures, and what are their historical links?

Question Four: Where are the major clusters of greenspaces located in Edinburgh?

What common features or characteristics do these greenspaces have?

Question Five: What is the precise location of Newbigging in Angus? Can you describe any landmarks or features nearby?

Question Six: How was Arthur's Seat formed? What significant geological events have impacted it over time?

```

1 Appendix A: Web Scraping Gazetteer for Scotland data into PostgreSQL database
2
3 #Inspired by this website: https://medium.com/@sangitaaryans/a-complete-guide-to-
  web-scraping-with-selenium-python-in-2024-4c856890be68
4
5 import os #Allows interact with computer system
6 import re #Allows relevant information to be retrieved.
7 import psycopg2 #Connects to PostgreSQL database.
8 from psycopg2 import pool #Allows multiple connections to database
9
10 #Selenium functions
11 from selenium import webdriver
12 from selenium.webdriver.common.by import By
13 from selenium.webdriver.chrome.service import Service
14 from selenium.webdriver.support.ui import WebDriverWait
15 from selenium.webdriver.support import expected_conditions as EC
16 from selenium.webdriver.chrome.options import Options
17 from webdriver_manager.chrome import ChromeDriverManager
18
19 # Sets up PostgreSQL database connection with environment variables
20 connection_pool = psycopg2.pool.SimpleConnectionPool(
21     minconn=1,
22     maxconn=10,
23     dbname=os.getenv('DB_NAME'),
24     user=os.getenv('DB_USER'),
25     host=os.getenv('DB_HOST'),
26     password=os.getenv('DB_PASS')
27 )
28
29 def clean_text(text):
30     '''Removes unwanted characters and splits paragraphs'''
31     return ' '.join(text.split())
32
33 def selenium_scrape_and_store(place_name):
34     '''Carries out web scraping by using Selenium'''
35     #Goes to initial Gazetteer For Scotland URL
36     url = 'https://www.scottish-places.info/anyword.html'
37     options = Options()
38     #Install Chrome Driver Manager- this means that when this script is run,
    chrome automatically open so it is visible how Selenium navigates the website for
    debugging purposes.
39     driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()),
    options=options)
40     #Get url of initial webpage.
41     driver.get(url)
42
43     try: #If website is successfully retrieved.
44         #Waits 20 seconds to avoid selenium timing out if the button is not
    immediately located.
45         search_box = WebDriverWait(driver, 20).until(
46             EC.presence_of_element_located((By.NAME, 'NAME'))) #Locates Search Box

```

```

47         )
48         search_box.send_keys(place_name)#Enters place name into search box.
49
50         submit_button = WebDriverWait(driver, 20).until( #Submits place name into
search box.
51             EC.element_to_be_clickable((By.XPATH, "///input[@type='submit' and
@value='Search']"))
52         )
53         submit_button.click()#Clicks Submit Button.
54
55         place_link = WebDriverWait(driver, 20).until(#Searches on Settlement
Place Link specifically.
56             EC.element_to_be_clickable((By.XPATH, f"///a[contains(@href,
'/towns/') and contains(translate(text(), 'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
'abcdefghijklmnopqrstuvwxyz'), '{place_name.lower()}')]"))
57         )
58         #Saves link for storage in PostgreSQL database.
59         link = place_link.get_attribute('href')
60         #Clicks on place link.
61         place_link.click()
62         #Finds place name in webpage.
63         place_name_elem = driver.find_element(By.XPATH,
"/html/body/table/tbody/tr/td[2]/table/tbody/tr[1]/td/blockquote/div/h1")
64         place_name = clean_text(place_name_elem.text)#Cleans and save place name.
65
66         #Finds description
67         description_elem = driver.find_element(By.XPATH,
"/html/body/table/tbody/tr/td[2]/table/tbody/tr[1]/td/blockquote/div/div[2]")
68         description = clean_text(description_elem.text) #Cleans and saves
description.
69
70         #Finds more details icon.
71         more_details_icon = WebDriverWait(driver, 20).until(
72             EC.element_to_be_clickable((By.XPATH, "///img[@title='More
Details']"))
73         )
74         more_details_icon.click() #Clicks on more details icon.
75
76         #Finds latitude information in more details webpage.
77         latitude_text = WebDriverWait(driver, 20).until(
78             EC.presence_of_element_located((By.XPATH,
"/td[@class='left_text']/b[contains(text(), '°N') or contains(text(), '°S')]"))
79         ).text.strip()
80
81         #Finds longitude information in more details webpage.
82         longitude_text = WebDriverWait(driver, 20).until(
83             EC.presence_of_element_located((By.XPATH,
"/td[@class='left_text']/b[contains(text(), '°E') or contains(text(), '°W')]"))
84         ).text.strip()
85
86         # Convert coordinates to decimal
87         latitude = convert_to_decimal(latitude_text)

```

```

88     longitude = convert_to_decimal(longitude_text)
89
90     #Finds geographical Information
91     geography_elem = driver.find_element(By.XPATH,
92     "/html/body/table/tbody/tr/td[2]/table/tbody/tr[1]/td/blockquote/div/div/div[1]")
93     geography = clean_text(geography_elem.text)
94
95     full_description = f"{description}\n\n{geography}" #Amends extra
geography information to description.
96
97     # Inserts data into the database
98     conn = connection_pool.getconn()
99     if conn:
100         cursor = conn.cursor()
101         insert_query = """
102             INSERT INTO millie.edinburgh1 (place_name, description, latitude,
longitude, link)
103             VALUES (%s, %s, %s, %s, %s)
104             """
105         cursor.execute(insert_query, (place_name, full_description, latitude,
longitude, link))
106         conn.commit()
107         cursor.close() #Closes connection to database.
108         connection_pool.putconn(conn)
109
110     except Exception as e: #error message if link could not be accessed.
111         print(f"Error accessing link for {place_name}: {str(e)}")
112
113     finally:
114         driver.quit() #Quit connection
115
116 import re
117
118 def convert_to_decimal(coord_text):
119     """Converts coordinates to decimal."""
120     try:
121         parts = re.split('[°]', coord_text) #Split coordinates into latitude and
longitude.
122         degree = float(parts[0].strip()) #Converts to float and strips white
space..
123         direction = parts[1].strip()
124
125         if direction in ['S', 'W']: #if coordinate is South or West
126             degree = -degree #Convert to negative to match british coordinates
127         return degree
128     except (ValueError, IndexError) as e: #error message
129         print(f"Error converting coordinate '{coord_text}': {str(e)}")
130         return None
131
132 #Usage example
133 selenium_scrape_and_store("Edinburgh")

```

```

1 Appendix B: Embedding PostgreSQL Database Script
2 import os #module for interacting with operating system.
3 import openai #imports OpenAI functionality.
4 import psycopg2 #enables connections to postgresQL database.
5 from dotenv import load_dotenv #loads testing environment.
6
7 load_dotenv() #This is for testing purposes onnly
8
9 openai.api_key = os.getenv("OPENAI_API_KEY") #Retrieves OpenAI Environment
  variable.
10
11 def get_embedding(text):
12     '''Creates embedding of the PostgreSQL Information'''
13     if text is None or text.strip() == '': #Replaces None or empty text with a
  single space to prevent errors.
14         text = " "
15     response = openai.Embedding.create(
16         model="text-embedding-ada-002", #Defines OpenAI embedding model.
17         input=[text]
18     )
19     return response['data'][0]['embedding'] #Returns embedded data.
20
21 def connect_to_db(): #Connects to database (should be set to environment
  variables)
22     return psycopg2.connect(
23         dbname='redacted',
24         user='redacted',
25         host='redacted',
26     )
27
28 def create_backup_table(cursor):
29     ''' Creates a backup table to prevent data loss.'''
30     #Creates a PostgreSQL backup table
31     #Structure is copied, but no data is included initially
32     cursor.execute("""
33     CREATE TABLE IF NOT EXISTS millie.chatbot_backup10 AS #Creates a PostgreSQL
  backup table
34     TABLE millie.chatbot WITH NO DATA;
35     """)
36     cursor.execute("""
37     INSERT INTO millie.chatbot_backup10
38     SELECT * FROM millie.chatbot;
39     """)
40
41 def create_vector_columns(cursor):
42     ''' Creates vector columns in PostgreSQL database for embedded data to go
  into.'''
43     #Alters table to add vector column if they do not already exist.
44     cursor.execute("""
45     ALTER TABLE millie.chatbot
46     ADD COLUMN IF NOT EXISTS place_name_vector VECTOR;

```

```

47     """
48     cursor.execute("""
49     ALTER TABLE millie.chatbot
50     ADD COLUMN IF NOT EXISTS latitude_vector VECTOR;
51     """)
52     cursor.execute("""
53     ALTER TABLE millie.chatbot
54     ADD COLUMN IF NOT EXISTS longitude_vector VECTOR;
55     """)
56     cursor.execute("""
57     ALTER TABLE millie.chatbot
58     ADD COLUMN IF NOT EXISTS description_vector VECTOR;
59     """)
60     cursor.execute("""
61     ALTER TABLE millie.chatbot
62     ADD COLUMN IF NOT EXISTS link_vector VECTOR;
63     """)
64
65 def vectorize_and_store():
66     '''Vectorises/embeds data and stores it back in postgresQL database'''
67     conn = connect_to_db() #Connects to PostgreSQL Database.
68     cursor = conn.cursor()
69
70     create_backup_table(cursor) #Creates backup table.
71     create_vector_columns(cursor) #creates vector columns.
72     #Selects relevant columns from database, including vector columns where the
data has not already been converted.
73     cursor.execute("""
74     SELECT row_id, place_name, latitude, longitude, description, link
75     FROM millie.chatbot
76     WHERE place_name_vector IS NULL
77         OR latitude_vector IS NULL
78         OR longitude_vector IS NULL
79         OR description_vector IS NULL
80         OR link_vector IS NULL;
81     """)
82     rows = cursor.fetchall() #fetches all rows.
83     #Prints statement for debudding.
84     print(f"Fetchd {len(rows)} rows for processing.")
85
86     #For each row
87     for row in rows:
88         row_id, place_name, latitude, longitude, description, link = row
#Defines what is included in each row.
89
90         description = description or " " #Description or empty space.
91         print(f"Processing row ID: {row_id}") #Prints statement for debugging.
92
93         #Creates embedding for each column (and each row)
94         place_name_embedding = get_embedding(place_name)
95         latitude_embedding = get_embedding(latitude)

```

```
96     longitude_embedding = get_embedding(longitude)
97     description_embedding = get_embedding(description)
98     link_embedding = get_embedding(link)
99
100     #Updates the PostgreSQL table with the new information.
101     cursor.execute("""
102     UPDATE millie.chatbot
103     SET place_name_vector = %s,
104         description_vector = %s,
105         latitude_vector = %s,
106         longitude_vector = %s,
107         link_vector = %s
108     WHERE row_id = %s
109     """, (place_name_embedding, description_embedding, latitude_embedding,
longitude_embedding, link_embedding, row_id))
110     #prints statement for debugging.
111     print(f"Updated row ID: {row_id} with embeddings.")
112     #Closes database connection.
113     conn.commit()
114     cursor.close()
115     conn.close()
116
117 #runs application.
118 if __name__ == "__main__":
119     vectorize_and_store()
```

```

1 Appendix C: GeoBot Flask App Python Script
2
3 ''' Firstly, the external packages used are imported. The import function allows the use of external modules
  in the app. These were first downloaded onto the computer then imported into this script'''
4
5 #GeoBot Flask App Imports
6 from flask import Flask, jsonify, request, render_template, redirect, url_for, session #Web framework for
  building applications.
7 from flask_cors import CORS #Security Feature to define which domains are allowed to access resources on the
  server.
8 import os #Module for interacting with the operating system.
9 import openai #Library to access OpenAI's capabilities.
10
11 #PostgreSQL Database Imports
12 import psycopg2 #Allows connections to PostgreSQL databases.
13 from psycopg2 import pool #Module for creating a pool of connections to a PostgreSQL database, allowing
  multiple users.
14 import psycopg2.extras #Provides additional database functionalities.
15
16 #Map Imports
17 import folium #Library for creating interactive maps.
18 import openrouteservice #Library for accessing route and navigation services.
19 from openrouteservice import convert #Used for converting data formats.
20 import re #Carries out regular expression matching operations, to find phrases that match a specified
  pattern.
21
22 #Multiple User Imports
23 import secrets #Module for generating secure tokens and random numbers.
24 import threading #Allows code to run at the same time.
25 import queue #Provides a safe way to exchange data between threads.
26
27 ''' The Flask app is then initialised so GeoBot can run and connect to its website.
  This involves setting up a session for each user and retrieving the connecting to the external PostgreSQL
  Database'''
28
29
30 app = Flask(__name__, template_folder='templates') #Initialising Flask app, specifying where the website
  code is found so the app can connect to it.
31 CORS(app) #Enables CORS for cross-origin requests (security measure).
32
33 password = 'GIS2025!' #Password for login page. This should be changed to an environment variable instead of
  being hardcoded into flask app.
34 app.config['SESSION_PERMANENT'] = False #Sets user session to expire when the browser closes.
35 app.secret_key = secrets.token_hex(16) #Generates a secure secret key for sessions (security measure).
36 ors_api_key = os.environ.get("ORS_API_KEY") #Retrieves the Open Route Service API Key for map functionality.
37 client = openrouteservice.Client(key=ors_api_key) #Create ORS client with API key (so ORS can be used).
38
39 # Setup a connection pool for PostgreSQL with environment variables for credentials (security measure)
40 connection_pool = psycopg2.pool.SimpleConnectionPool(
41     minconn=1, #Minimum amount of users at once.
42     maxconn=10, #Maximum amount of users at once.
43     #Logs in to GeoBot database.
44     dbname=os.getenv('DB_NAME'), #Database name.
45     user=os.getenv('DB_USER'), #Username.
46     host=os.getenv('DB_HOST'), #Geosciences host name.
47     password=os.getenv('DB_PASS') #PostgreSQL user password.
48 )
49
50 request_queue = queue.Queue() #To manage multiple requests without the app breaking down.
51
52 # This section manages the user connections and PostgreSQL database.
53 # Def functions are only carried out when they are called later in the python script.
54 def get_connection():
55     '''Retrieves a user connection from the PostgreSQL database connection pool.'''
56     return connection_pool.getconn()
57
58 def release_connection(conn):
59     '''Returns the user connection to the pool for reuse.'''

```

```

60     connection_pool.putconn(conn)
61
62 def close_all_connections():
63     '''Closed all connections in the pool.'''
64     connection_pool.closeall()
65
66 def get_query_vector(query_text):
67     '''Uses the OpenAI API embedding model to create an embedding for the query (turns query into numbers)
68     .'''
69     response = openai.Embedding.create(
70         input=[query_text], #query text is defined as input,
71         model="text-embedding-ada-002" #Specifies the model used for embedding.
72     )
73     return response['data'][0]['embedding'] #Returns the generated embedding.
74
75 def vector_search(query_vector):
76     '''This function connects to the PostgreSQL database and retrieves the most similar information to the
77     user query (similarity searching)'''
78     conn = get_connection() #Gets a database connection from the pool.
79     try: #only runs if connection is found.
80         cursor = conn.cursor() #creates a cursor object so the query can be executed.
81         query_vector = list(query_vector) #Converts the query vector to a list so it can be compared to
82         database.
83         #Calculates the smallest distance between the query vector and each vector column (description,
84         place name, latitude, longitude, link) using euclidean similarity.
85         # This distance is labeled as distance. FROM selects the table, ORDER By orders the retrived
86         information by distance (how far away the information is from user query) and LIMITS means that only 10 rows
87         of the most relevant information are returned to conserve running costs and efficiency.
88         select_query = """
89         SELECT place_name, description, latitude, longitude, link,
90         LEAST(
91             description_vector <-> %s::vector,
92             place_name_vector <-> %s::vector,
93             latitude_vector <-> %s::vector,
94             longitude_vector <-> %s::vector,
95             link_vector <-> %s::vector
96         ) AS distance
97         FROM millie.chatbot
98         ORDER BY distance
99         LIMIT 10;
100        """
101         cursor.execute(select_query, (query_vector,) * 5) #Executes the SQL query, substituting the query
102         vector into each placeholder five times (parametized queries, security measure).
103         results = cursor.fetchall() #Fetches all results from the query execution.
104         cursor.close() #Closes connection.
105         return results #Returns results.
106     finally:
107         release_connection(conn) #Ensures the database connection is returned to the pool, even if an error
108         occurs.
109
110 # This section deals with the map functionality.
111 def extract_place_names(text):
112     ''' Extracts place names from the response. GeoBot is trained to always produce responses in a certain
113     format so thos can be retrieved.'''
114     pattern = r"(?<=Coordinates for )([\w\s'\-]+)" #Regex module finds place names after "Coordinates for".
115     matches = re.findall(pattern, text) #Finds all matches in the text.
116     return [match.strip() for match in matches] #Strips white space of result.
117
118 def extract_coordinates(text):
119     ''' Extracts coordinates from response.'''
120     pattern = r"Latitude:\s+([+-]?[0-9]*\.[0-9]+),\s+Longitude:\s+([+-]?[0-9]*\.[0-9]+)" #Extracts
121     coordinates.
122     matches = re.findall(pattern, text) #Finds all matches in the text.
123     coordinates = [(float(lat), float(lon)) for lat, lon in matches] #Converts matches to tuples of floats
124     so it can be used to plot the map markers.
125     return coordinates #return results.
126
127 def convert_links_to_html(text):

```

```

117     '''Converts links in response to HTML to create hyperlinks'''
118     url_pattern = r'\[((^|)]+)\)\((https?:\/\/[^\s]+)\)' #Identifies URL Pattern
119     return re.sub(url_pattern, r'<a href="\2">\1</a>', text) #Converts to HTML hyperlink.
120
121 def ensure_negative_longitude(lon):
122     '''As GeoBot focuses on Scotland, this ensures that all coordinates match the British format.'''
123     return -abs(lon)
124
125 def get_route(client, start, end, profile='driving-car'):
126     '''Calculates routes between each location on the leaflet map.'''
127     try: #only runs if successful route has been found between a set of markers, if not then skips to the
next pair to prevent app crashing.
128         coords = [start, end] #Gets start and coordinates between each marker.
129         route = client.directions(coords, profile=profile) #Requests route directions from Open Route
Service.
130
131         geometry = route['routes'][0]['geometry'] #Extracts the route geometry (the route line)
132         decoded = convert.decode_polyline(geometry) #Converts geometry into a polyline.
133         distance_miles = route['routes'][0]['summary']['distance'] * 0.000621371 #Converts distance to
miles.
134
135         return [(point[1], point[0]) for point in decoded['coordinates']], distance_miles #Returns finalised
coordinates and calculates the distance.
136     except openrouteservice.exceptions.ApiError as e: #except if error has been found.
137         print(f"Error retrieving route: {e}") #print error message in console.
138         return None, None #Return no routes if error occurs.
139
140 def plot_map(coordinates, place_names, ors_api_key):
141     '''Once all the information has been retrieved from response, this creates the map.'''
142     coordinates = [(ensure_negative_longitude(lon), lat) for lat, lon in coordinates] #Ensures all
coordinates are in British format.
143     #Calculates the average coordinates for centering the map.
144     avg_lat = sum(lat for _, lat in coordinates) / len(coordinates)
145     avg_lon = sum(lon for lon, _ in coordinates) / len(coordinates)
146
147     #Creates a Folium map centered on the average coordinates.
148     my_map = folium.Map(
149         location=[avg_lat, avg_lon], #sets location.
150         zoom_start=13, #Zoom is set to 13 for readability.
151         width='100%', #Width and height are set to maximum.
152         height='100vh'
153     )
154
155     #Adds a transport tile layer to the map.
156     folium.TileLayer(
157         tiles='https://tile.memomaps.de/tilegen/{z}/{x}/{y}.png', #Memomaps tile.
158         attr='Map data &copy; OpenStreetMap contributors',
159         name='Transport Options', #Sets layer name.
160         show=False, #Initially layer is not set to on in map.
161         overlay=True #Sets it as overlay.
162     ).add_to(my_map) #Adds to map.
163
164     #Adds a satellite tile layer to the map.
165     folium.TileLayer(
166         tiles='https://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer/tile/{z}/{y}/{x}', #ArcGIS
online satellite tile layer.
167         attr=('Tiles &copy; Esri &mdash; Source: Esri, Maxar, Earthstar Geographics, '
168             'and the GIS User Community'),
169         name='Satellite',
170         overlay=False,
171         show=False
172     ).add_to(my_map)
173
174     #For every coordinate pair in response, create a green marker and add to map.
175     for (lon, lat), place in zip(coordinates, place_names):
176         folium.Marker(
177             location=[lat, lon],
178             popup=place,
179             icon=folium.Icon(color='green', icon='info-sign') #Had to use american spelling of colour for

```

```

Folium to work.
178     ).add_to(my_map)
179
180     #Add routes for cycling, car and walking and determine the colour.
181     for route_type, color, profile in [
182         ('Car Routes', 'red', 'driving-car'),
183         ('Walking Routes', 'blue', 'foot-walking'),
184         ('Cycling Routes', 'green', 'cycling-regular')
185     ]:
186         route_layer = folium.FeatureGroup(name=route_type) #Creates a feature group for each route type.
187         for i in range(len(coordinates) - 1): #Creates a loop that iterates over pairs of consecutive
coordinates.
188             route, distance_miles = get_route(client, coordinates[i], coordinates[i + 1], profile=profile)
#Retrieves route for each coordinate pair.
189             if route: #If a route is found.
190                 folium.PolyLine(
191                     route, #sets route.
192                     color=color, #sets colour
193                     weight=5, #sets thickness of line.
194                     popup=f"{distance_miles:.2f} miles by {route_type.split()[0]}"#Creates popup of
information of user clicks on it.
195                 ).add_to(route_layer) #Add to route layer and map.
196                 route_layer.add_to(my_map)
197
198     from folium.plugins import MeasureControl #Imports measuring feature to map from folium features.
199     my_map.add_child(MeasureControl())
200
201     folium.LayerControl().add_to(my_map)#Adds measuring tool to map.
202
203     return my_map._repr_html_() #Returns my map as a HTML feature so it can be embedded in website.
204
205 #This section deals with the flask app processes to construct a response through ELM (OpenAI GPT-4), drawing
upon the earlier functions.
206 def AskELM(input_text, conversation_history):
207     """
208     This function processes input text to response using OpenAI's GPT-4 model.
209
210     1. Converts the input text into a query vector.
211     2. Finds similar documents to query.
212     3. Constructs context parts from the database documents, including place names, descriptions, coordinates,
and links.
213     4. Sends this to OpenAI's GPT-4 for response generation.
214     5. Collects and formats the response, extracting place names and coordinates.
215     6. Returns the response text to the website.
216     """
217     query_vector = get_query_vector(input_text) #Process user query.
218     similar_docs = vector_search(query_vector) #Similarity seaches user query with database information.
219
220     place_names = [row[0] for row in similar_docs] #Extracts place names from similar documents.
221     context_parts = [] #Creates and sets the context list to empty. The context is essential so the LLM can
compare the vectorised information with the original text information- so it can understand the information.
222     for row in similar_docs: #For each row in the retrieved information, retrieves the specified information
unless it is not there then say that it is unknown or not available.
223         place_name = row[0] if len(row) > 0 else "Unknown"
224         description = row[1] if len(row) > 1 else "No description"
225         latitude = row[2] if len(row) > 2 else "Unknown"
226         longitude = row[3] if len(row) > 3 else "Unknown"
227         link = row[4] if len(row) > 4 else "No link available"
228         #Adds these details to the context.
229         context_parts.append(
230             f"Place Name: {place_name}\nDescription: {description}\nCoordinates: Latitude: {latitude},
Longitude: {longitude}\nLink: {link}"
231         )
232     #Join the different part of the context together.
233     context = ", ".join(context_parts)
234     #Creates and sets response text as empty.
235     response_text = ""
236     #Print the context (for checking that all is well in console.)

```

```

237     print('Context:', context)
238     #Calls upon the OpenAI's model to construct a response with context and system instructions.
239     stream = openai.ChatCompletion.create(messages=[
240         {"role": "system", "content":
241 "You are a geographical Retrieval augmented generation (RAG) chatbot and formatting assistant. Answer any
question, but You know about Edinburgh Greenspaces, any other knowledge is from your base OpenAI knowledge
and not from M.I.L.L.I.E database information. "
242 "Always format the information as follows, using UK English Spelling:\n\n"
243 "1. Provide a detailed description.\n"
244 "2. In every response include and List the coordinates and link for every place name in this specific
format: Link must be same as place name.\n"
245 " - Coordinates for [Place Name]: Latitude: [Latitude], Longitude:-[Longitude]. Follow this [link] for
more information about Place Name. Only attach Gazetteer for Scotland Links.\n\n"
246 "3. If providing a distance calculation, use this simplified formula for straight-line distances: \n"
247 "d =  $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ . Convert this to miles. Do not explain the methodology."
248 "Note: The chatbot's map function provides distance calculations that consider local
infrastructure.\n\n"
249     },
250     {"role": "user", "content": f"Context: {context}\nQuestion: {input_text}"} #Adds the context and
input text to conversation history.
251     ] + conversation_history,
252     model="gpt-4-turbo", stream=True) #Sets the GPT-4 Model and stream=true allows the response to be
generated as it is being processed.
253     #Collects the response text.
254     for chunk in stream:
255         chunk_content = chunk.choices[0].delta.get('content', '') #Extracts content from the current
response chunk.
256         response_text += chunk_content #Adds chunk to full response text.
257         print(response_text)#prints for debugging.
258         response_text = convert_links_to_html(response_text)#converts response links to hyperlinks.
259         place_names = extract_place_names(response_text)#extracts place names for map.
260         coordinates = extract_coordinates(response_text)#Extracts coordinates for map.
261         return response_text, similar_docs, place_names, coordinates #Returns response text, context, place
names and coordinates.
262
263 def process_queue():
264     '''This function handles the queue of users.'''
265     while True: #While there is a queue.
266         func, args = request_queue.get() # Takes the next task.
267         func(*args) # Does the task.
268         request_queue.task_done() # Marks the task as finished.
269
270 threading.Thread(target=process_queue, daemon=True).start()#Processes the queue using threads.
271
272 def ask_elm_and_respond(input_text, conversation_history, response_callback):
273     '''This ensures a response is created by ELM (OpenAI GPT-4)'''
274     try:
275         #Calls AskELM to process the input text and get relevant data.
276         response_text, similar_docs, place_names, coordinates = AskELM(input_text, conversation_history)
277         #Returns the response data to create a response.
278         response_callback({'response_text': response_text, 'place_names': place_names, 'coordinates':
coordinates})
279         #Except if an error occurs (signalling GeoBot has crashed.)
280     except Exception as e:
281         #Prints error in console.
282         print(f"Error: {e}")
283         response_callback({'error': 'An error occurred.'})
284
285 #This final section sets up Flask routes to manage user authentication and interactions with the GeoBot web
application.
286 @app.route('/', methods=['GET', 'POST'])
287 #When the user clicks on the GeoBot website, the flask app sends them to the login page '/'.
288 def login():
289     '''This initialises the login html page'''
290     error = ''
291     if request.method == 'POST':
292         password = request.form.get('password') #Retrieves the password.
293         if password == 'GIS2025!': #If password is correct.

```

```

294         session['logged_in'] = True #Log the user in.
295         return redirect(url_for('main')) #irect to main webpage.
296     else: #If password is not correct.
297         error = 'Incorrect password, please try again.' #Present error message
298         return render_template('login.html', error=error) #Intalise login webpage.
299
300 @app.route('/main')
301 #If user logs in correctly, they are then sent to the main webpage.
302 def main():
303     '''Handles main webpage functionality.'''
304     if not session.get('logged_in'): #If not logged in, send to login page.
305         return redirect(url_for('login'))
306     #Create conversation history if it does not exist and display welcome message.
307     if 'conversation_history' not in session:
308         session['conversation_history'] = [{'role': 'assistant', 'content': "Welcome to GeoBot! I know about
Edinburgh Greenspaces and use Gazetteer for Scotland information to inform my responses. How can I assist
you today?"}]
309     return render_template('index2.html', conversation_history=session['conversation_history']) #Returns
main webpage 'index2.html'.
310
311 @app.route('/logout')
312 #If logout button is pressed, Clears the session and redirects to login.
313 def logout():
314     session.pop('conversation_history', None) #Clears conversation history.
315     session.clear() #Clears session data.
316     return redirect(url_for('login')) #Redirects to login page.
317
318 @app.route('/generate_map', methods=['POST'])
319 #If generate map button is pressed, create map.
320 def generate_map():
321     '''generates map'''
322     data = request.json #Requests json map data.
323     coordinates = data.get('coordinates', []) #Retrieves collected coordinate data.
324     place_names = data.get('place_names', []) #Retrieves collected place name data.
325     map_html = plot_map(coordinates, place_names, ors_api_key) #Plots map.
326     return jsonify({'map_html': map_html}) #Returns map for display on HTML Website.
327
328 @app.route('/AskELM', methods=['POST'])
329 # Handles requests to AskELM, updating conversation history.
330 def handle_ask_elm():
331     ''' Ensures responses are created.'''
332     if not session.get('logged_in'): #If sessin is not created, redirect to login page.
333         return redirect(url_for('login'))
334     data = request.json #get JSON Data from request.
335     input_text = data.get('input_text', '') #Extract user query.
336     conversation_history = session.get('conversation_history', []) #get session conversation history.
337     response = {} #Set response.
338
339     conversation_history.append({'role': 'user', 'content': input_text}) #Append conversation history with
user query.
340     #Adds task to queue: process input with ask_elm_and_respond and update response.
341     request_queue.put((ask_elm_and_respond, (input_text, conversation_history, lambda res:
response.update(res))))
342     request_queue.join() #Waits for the queue to process request. Note: This is only suitable for a small
amount of users, for commercial applications this does not work well as GeoBot would keep breaking.
343
344     if 'response_text' in response:
345         conversation_history.append({'role': 'assistant', 'content': 'response_text'}) #Adds response to
converntion history.
346
347     session['conversation_history'] = conversation_history #Adds new conversation history to session.
348
349     return jsonify(response) #Returns and displays response.
350
351 if __name__ == '__main__':
352     app.run(debug=True) #Runs the flask app.

```

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25
- 26
- 27
- 28
- 29
- 30
- 31
- 32
- 33
- 34
- 35
- 36
- 37
- 38
- 39
- 40
- 41
- 42
- 43
- 44
- 45
- 46
- 47
- 48
- 49
- 50
- 51

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Secure Login</title>
  <!-- CSS Styling -->
  <style>
    body {
      font-family: Arial, sans-serif;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      margin: 0;
      background-color: #e9ecef;
    }
    #login-form {
      text-align: center;
      background-color: #ffffff;
      padding: 40px;
      border-radius: 12px;
      box-shadow: 0 6px 12px rgba(0, 0, 0, 0.15);
      max-width: 320px;
    }
    input[type="password"] {
      width: 100%;
      padding: 10px;
      margin: 20px 0;
      border: 1px solid #ced4da;
      border-radius: 4px;
      font-size: 16px;
    }
    button {
      background-color: #007bff;
      color: #fff;
      border: none;
      padding: 10px 20px;
      border-radius: 4px;
      font-size: 16px;
      cursor: pointer;
      transition: background-color 0.3s;
    }
    button:hover {
      background-color: #0056b3;
    }
    p {
      font-size: 14px;
      color: #6c757d;
    }
  </style>
</head>
```

```
52         .error {
53             color: #dc3545;
54         }
55     </style>
56 </head>
57 <body>
58     <div id="login-form">
59         <h2>Login</h2> <!-- Form title -->
60         <form method="POST">
61             <input type="password" name="password" placeholder="Enter Password"
required/> <!-- Password input section -->
62             <button type="submit">Enter</button> <!-- Form submit button -->
63         </form>
64         <p class="error">{{ error }}</p> <!-- Placeholder for error messages -->
65         <p>This chatbot specializes in Greenspaces in Edinburgh. For questions
outside this scope, it uses general knowledge that may not be accurate.</p> <!--
Text about the GeoBot -->
66     </div>
67 </body>
68 </html>
```

1 Appendix E: Main GeoBot HTML Webpage

```
2
3 <!DOCTYPE html>
4 <html lang="en">
5 <head>
6   <meta charset="UTF-8">
7   <meta name="viewport" content="width=device-width, initial-scale=1.0">
8   <title>GeoBot
9   </title>
10  <link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css" />
11  <style>
12    /*CSS Styling- Formats the look of GeoBot */
13    body {
14      font-family: 'Inter', sans-serif;
15      margin: 0;
16      padding: 0;
17      display: flex;
18      justify-content: center;
19      align-items: center;
20      flex-direction: column;
21      height: 100vh;
22      background-color: #f0f0f0;
23      color: #111;
24    }
25    #instructions {
26      font-size: 1em;
27      display: none;
28      background-color: #fff;
29      color: #222;
30      padding: 20px;
31      border-radius: 12px;
32      width: 85vw;
33      box-shadow: 0 4px 6px rgba(0, 0, 0, 0.15);
34      text-align: center;
35      font-weight: 500;
36      line-height: 1.5;
37    }
38    #chat-container {
39      display: flex;
40      width: 95vw;
41      height: 80vh;
42      border-radius: 16px;
43      overflow: hidden;
44      box-shadow: 0 6px 12px rgba(0, 0, 0, 0.15);
45      background-color: #fff;
46      position: relative;
47    }
48    #chat-area, #map-container {
49      height: 100%;
50      display: flex;
51      flex-direction: column;
```

```
52     }
53     #chat-area {
54         width: 50%;
55         border-right: 1px solid #ccc;
56     }
57     #map-container {
58         width: 50%;
59     }
60     #chat-header, #map-header {
61         background-color: #fff;
62         color: #111;
63         padding: 16px 24px;
64         text-align: center;
65         font-size: 1.7rem;
66         font-weight: 600;
67         position: relative;
68         border-bottom: 1px solid #d0d0d0;
69     }
70     #map {
71         width: 100%;
72         height: 100%;
73     }
74     .header-button {
75         position: absolute;
76         top: 14px;
77         background: none;
78         border: none;
79         width: 30px;
80         height: 30px;
81         font-size: 1.5em;
82         cursor: pointer;
83         color: #111;
84         transition: color 0.3s;
85         display: flex;
86         flex-direction: column;
87         align-items: center;
88     }
89     .header-button:hover {
90         color: #555;
91     }
92     #logout {
93         left: 10px;
94     }
95     #info-button {
96         right: 10px;
97     }
98     .header-button span {
99         font-size: 0.8rem;
100        margin-top: 2px;
101    }
102    #chat-box {
```

```
103         flex-grow: 1;
104         padding: 24px;
105         overflow-y: auto;
106         display: flex;
107         flex-direction: column;
108         background-color: #f0f0f0;
109     }
110     #chat-input-area {
111         display: flex;
112         padding: 16px;
113         background-color: #fff;
114         border-top: 1px solid #ccc;
115     }
116     #chat-input {
117         flex-grow: 1;
118         padding: 12px 16px;
119         font-size: 1.4rem;
120         border: 1px solid #777;
121         border-radius: 8px;
122         outline: none;
123         background-color: #fff;
124     }
125     #send-button, #map-button {
126         margin-left: 12px;
127         padding: 12px 16px;
128         font-size: 1.4rem;
129         background-color: #0d8c6f;
130         color: #fff;
131         border: none;
132         border-radius: 8px;
133         cursor: pointer;
134         transition: background-color 0.3s;
135     }
136     #send-button:hover, #map-button:hover {
137         background-color: #0b7b60;
138     }
139     #map-button {
140         display: none;
141     }
142     .user-message, .bot-message {
143         margin: 12px 0;
144         padding: 16px 20px;
145         border-radius: 16px;
146         max-width: 80%;
147         font-size: 1.4rem;
148         line-height: 1.5;
149         white-space: pre-wrap;
150         word-break: break-word;
151         box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
152     }
153     .user-message {
```

```

154         background-color: #b2e0ff;
155         align-self: flex-end;
156         text-align: right;
157     }
158     .bot-message {
159         background-color: #d8d8d8;
160         align-self: flex-start;
161         text-align: left;
162     }
163     .typing-indicator {
164         display: inline-block;
165         width: 8px;
166         height: 8px;
167         background-color: #666;
168         border-radius: 50%;
169         margin: 0 2px;
170         animation: blink 1.4s infinite both;
171     }
172     .typing-indicator:nth-child(2) {
173         animation-delay: 0.2s;
174     }
175     .typing-indicator:nth-child(3) {
176         animation-delay: 0.4s;
177     } /* Inspired by this website: https://www.includehelp.com/html/css-
loading-animations.aspx/*
178     @keyframes blink {
179         0%, 100% { opacity: 0.2; }
180         50% { opacity: 1; }
181     }
182     #resizer {
183         width: 30px;
184         cursor: ew-resize;
185         position: absolute;
186         top: 0;
187         bottom: 0;
188         left: calc(50% - 15px);
189         z-index: 1000;
190         background-image:
url('https://cdn3.iconfinder.com/data/icons/cursor-
16x16/16/resize_width_cursor_arrow-512.png');
191         background-size: contain;
192         background-repeat: no-repeat;
193         background-position: center;
194     }
195     </style>
196     <!-- HTML for the chat interface and map functionality -->
197 </head>
198 <body>
199     <div id="chat-container">
200         <!-- Container for the chat interface -->
201         <div id="chat-area">

```

```

202         <!-- Area where chat is displayed -->
203         <div id="chat-header">
204             <!-- Header for chat, displays GeoBot and buttons -->
205             GeoBot
206             <button id="logout" class="header-button">
207                 &#x2716;
208                 <span>Logout</span> <!-- Logout button -->
209             </button>
210             <button id="info-button" class="header-button">
211                 i
212                 <span>Help</span> <!-- Information button -->
213             </button>
214         </div>
215         <div id="chat-box"></div> <!-- Container for chat messages -->
216         <div id="chat-input-area">
217             <input type="text" id="chat-input" placeholder="Type a
message..." /> <!-- User Input Placeholder -->
218             <button id="send-button">Send</button> <!-- Send button -->
219             <button id="map-button">Create Map</button> <!-- Create map
button -->
220         </div>
221     </div>
222     <div id="resizer"></div>
223     <div id="map-container"> <!-- Divider used for resizing the chat and
map area -->
224         <div id="map-header">
225             Map Generation <!-- Map Box Title -->
226         </div>
227         <div id="map"></div>
228     </div>
229 </div>
230 <div id="instructions"> <!-- Help Button Instructions -->
231     <p>Type your messages in the input box and press "Send". If chatbot
does not respond, refresh the page. If the problem continues, contact the
chatbot owner.</p>
232     <p>This chatbot knows about Edinburgh Greenspaces, with information
sourced from the Gazetteer for Scotland. It will answer other queries but this
may be inaccurate. View map-related data via the "View Map" button when
available. This map shows the locations mentioned, as well as dynamic transport
routing. Change the layers through the button on the top right corner of the
map.</p>
233     <p>Some recommended questions to ask are:</p>
234     <p>Where is Newbigging in Angus?<br>
235     What is the distance between Arthur's Seat and the Meadows?<br>
236     What are the greenspaces in Edinburgh associated with famous people?
<br>
237     Are there any clusters of greenspaces in Edinburgh?<br>
238     What are the geographical features of greenspaces in Edinburgh?</p>
239 </div>
240 <script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>
241 <script>
242     /* JavaScript for chat and map functionality */

```

```

243     const chatBox = document.getElementById('chat-box');
244
245     function sendMessage(message, sender) {
246         // Function to send a GeoBot message to the chat box
247         const chatMessage = document.createElement('div');
248         chatMessage.classList.add(sender === 'user' ? 'user-message' :
'bot-message');
249         chatMessage.textContent = message;
250         chatBox.appendChild(chatMessage); // Adds GeoBot message to the
chat box
251         chatBox.scrollTop = chatBox.scrollHeight; // Enables scrolling
252     }
253     function loadConversationHistory(conversationHistory) {
254         // Loads past conversation history
255         conversationHistory.forEach(entry => {
256             sendMessage(entry.content, entry.role);
257         });
258     }
259     // Waits to load conversation history until DOM is loaded- this enables
the flask app to interact with this website.
260     window.addEventListener('DOMContentLoaded', () => {
261         const conversationHistory = [{ conversation_history|tojson }];
262         loadConversationHistory(conversationHistory);
263     });
264     // Loads past conversation history
265     const sendButton = document.getElementById('send-button'); //
References to send button so the buttons are clickable.
266     const mapButton = document.getElementById('map-button');
267     const chatInput = document.getElementById('chat-input');
268     const mapContainer = document.getElementById('map-container');
269     const chatArea = document.getElementById('chat-area');
270     const logoutButton = document.getElementById('logout');
271     const infoButton = document.getElementById('info-button');
272     const instructions = document.getElementById('instructions');
273     const resizer = document.getElementById('resizer');
274     const chatContainer = document.getElementById('chat-container');
275
276     const conversationHistory = []; // Tracks conversation history.
277
278     let currentMapData = null; //Sets initial map to null
279     let isDragging = false;
280     let map;
281     let popupShown = false;
282
283     const enableDragging = () => {
284         // Enables map to be moved around by user to make chatbox bigger.
285         document.body.style.cursor = 'ew-resize';
286         document.documentElement.style.userSelect = 'none';
287         if (map) map.dragging.disable();
288     };
289

```

```

290     const disableDragging = () => {
291         // disables dragging of map
292         isDragging = false;
293         document.body.style.cursor = 'default';
294         document.documentElement.style.userSelect = 'auto';
295         if (map) map.dragging.enable();
296     };
297
298     resizer.addEventListener('mousedown', (e) => {
299         e.preventDefault(); // When resizer button is first clicked show
popup message.
300         if (!popupShown) {
301             alert('To make the map smaller, when the resizer button is
activated move the map back by dragging on the bottom of the page.');
```

302 popupShown = true;

303 }

304 isDragging = true;

305 enableDragging();

306 });

307

308 document.addEventListener('mousemove', (e) => {

309 if (!isDragging) return;

310 e.preventDefault(); // Means that map cannot be clicked on during
drag to create a more seamless experience.

311 // Creates new width of map and chat boxes when the map is being
expanded or made smaller.

312 const containerRect = chatContainer.getBoundingClientRect();

313 const newX = e.clientX - containerRect.left;

314 const chatWidth = (newX / containerRect.width) * 100;

315 const mapWidth = 100 - chatWidth;

316

317 if (chatWidth > 10 && mapWidth > 10) {

318 chatArea.style.width = `\${chatWidth}%`;

319 mapContainer.style.width = `\${mapWidth}%`;

320 resizer.style.left = `\${newX - 15}px`;

321 }

322 });

323 // Once map has stayed at new size, disable the dragging movement.

324 document.addEventListener('mouseup', (e) => {

325 e.preventDefault();

326 disableDragging();

327 });

328

329 // Function to open and show map

330 function openMap() {

331 if (currentMapData) { // If there is map data available

332 mapContainer.style.display = 'flex'; //Show map

333 const { coordinates, placeNames } = currentMapData; // Open

Data

334

335 fetch(`\${url\_for('generate\_map')}`, { //Collect create map
from flask app.

```

336         method: 'POST',
337         headers: { 'Content-Type': 'application/json' },
338         body: JSON.stringify({ coordinates: coordinates,
place_names: placeNames })
339     })
340     .then(response => response.json())
341     .then(data => {
342         document.getElementById('map').innerHTML = data.map_html;
//Insert Initial Map HTML for when map of coordinates is created.
343         document.getElementById('map').style.height = '100%'; //
Set height as 100%
344         map = L.map('map').setView([55.9533, -3.1883], 13); // Set
view of map initally to Edinburgh until the new map is presented.
345         L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
346             maxZoom: 19
347         }).addTo(map);
348     })
349     .catch(error => console.error('Error:', error));
350 }
351 }
352
353     sendButton.addEventListener('click', () => {
354         // When send button is clicked.
355         const userMessage = chatInput.value;
356         if (userMessage.trim()) {
357             sendMessage(userMessage, 'user');
358             chatInput.value = '';
359             chatBox.scrollTop = chatBox.scrollHeight;
360
361             conversationHistory.push({ "role": "user", "content":
userMessage }); // Add user query to conversation history.
362             //Creates typing indicator when geobot is thinking- this
covers up the slow response time while the RAG Model retrieves information.
363             const botChat = document.createElement('div');
364             botChat.classList.add('bot-message');
365             botChat.innerHTML = 'GeoBot: <span class="typing-indicator">
</span><span class="typing-indicator"></span><span class="typing-indicator">
</span>';
366             chatBox.appendChild(botChat);
367             chatBox.scrollTop = chatBox.scrollHeight;
368             // Once the geoBot response has been created, the typing
indicator stops and a response is presented.
369             fetch("{ url_for('handle_ask_elm') }}", { // Sends user query
to geobot flask app to produce a response.
370                 method: 'POST',
371                 headers: { 'Content-Type': 'application/json' },
372                 body: JSON.stringify({ input_text: userMessage,
conversation_history: conversationHistory })
373             })
374             .then(response => response.json())
375             .then(data => {

```

```

376         const responseText = data.response_text.replace(/\*/g,
377         '').split('\n');
378         botChat.innerHTML = 'GeoBot: '; // typing indicator stops.
379         let i = 0;
380         const typingInterval = setInterval(() => {
381             if (i < responseText.length) {
382                 const paragraph = document.createElement('p'); //
383                 Creates paragraphs so response is not one block of text,
384                 paragraph.innerHTML = responseText[i];
385                 botChat.appendChild(paragraph);
386                 i++;
387                 chatBox.scrollTop = chatBox.scrollHeight;
388             } else {
389                 clearInterval(typingInterval);
390                 const coordinates = data.coordinates; // gets
391                 coordinates from response.
392                 const placeNames = data.place_names; // gets place
393                 names from response.
394                 if (coordinates && coordinates.length > 0) { // If
395                 there are coordinates, show display map button.
396                     currentMapData = { coordinates, placeNames };
397                     mapButton.style.display = 'inline-block';
398                 } else {
399                     mapButton.style.display = 'none';
400                     currentMapData = null; // if there are not
401                 coordinates, do not display create map button.
402                 }
403             }
404             }, 750);
405         })
406         .catch(error => {
407             console.error('Error:', error);
408             mapButton.style.display = 'none';
409             currentMapData = null;
410         });
411     }
412     });
413     mapButton.addEventListener('click', openMap);
414     logoutButton.addEventListener('click', () => { // if logout button is
415     pressed then logout user
416         window.location.href = "{{ url_for('logout') }}";
417     });
418     infoButton.addEventListener('click', () => {
419         instructions.style.display = instructions.style.display === 'none'
420         ? 'block' : 'none';
421     });
422     window.addEventListener('DOMContentLoaded', () => { // Initialise
423     initial Open Street map when website is loaded.

```

```
419         map = L.map('map').setView([55.9533, -3.1883], 13);
420         L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
421             maxZoom: 19
422         }).addTo(map);
423     });
424 </script>
425 </body>
426 </html>
```

```

1 Appendix F: Main Results Methodology Full Text Pairs Omitted For Readability
2
3 from collections import Counter #for counting words
4 import math #for mathematical functions
5 import re #imports re for keyword matching regular expressions
6 from rouge_score import rouge_scorer #imports rouge score calcuator
7 import textstat #imports textstat for readability scoring
8
9 # Defines stopwords (common words that are removed as they can impact results)
10 STOPWORDS = {"i", "me", "my", "myself", "we", "our", "ours", "ourselves",
    "you", "your", "yours", "yourself", "yourselves", "he", "him", "his",
    "himself", "she", "her", "hers", "herself", "it", "its", "itself", "they",
    "them", "their", "theirs", "themselves", "what", "which", "who", "whom",
    "this", "that", "these", "those", "am", "is", "are", "was", "were", "be",
    "been", "being", "have", "has", "had", "having", "do", "does", "did", "doing",
    "a", "an", "the", "and", "but", "if", "or", "because", "as", "until", "while",
    "of", "at", "by", "for", "with", "about", "against", "between", "into",
    "through", "during", "before", "after", "above", "below", "to", "from", "up",
    "down", "in", "out", "on", "off", "over", "under", "again", "further", "then",
    "once", "here", "there", "when", "where", "why", "how", "all", "any", "both",
    "each", "few", "more", "most", "other", "some", "such", "no", "nor", "not",
    "only", "own", "same", "so", "than", "too", "very", "s", "t", "can", "will",
    "just", "don", "should", "now"}
11
12 #Inspired by this website: https://pypi.org/project/textstat/
13 def average_reading_score(text_pairs, batch_size=6):
14     '''Calcuates the Flesch Reading score for each chatbot'''
15     #For every text pair in each batch (chatbot type)
16     for batch_start in range(0, len(text_pairs), batch_size):
17         #Select the current batch
18         batch = text_pairs[batch_start:batch_start + batch_size]
19         total_score = 0 #Create score section for batch
20         count = 0 #create count section for batch
21         #calculates the reading score of the response for each pair in each
    batch (chatbot type)
22         for i, (text1, _) in enumerate(batch, start=batch_start):
23             reading_score = textstat.flesch_reading_ease(text1)
24             total_score += reading_score #Adds score to total
25             count += 1
26
27         #Calculates and prints average score for the batch
28         average_score = total_score / count if count else 0
29         print(f"Batch Average Reading Score: {average_score}")
30
31 def preprocess(text):
32     '''Preprocesses text for analysis'''
33     #Removes common words from text
34     text = re.sub(r'\b(place name|coordinates|link)\b', '', text,
    flags=re.IGNORECASE)
35     #Removes whitespace
36     text = re.sub(r'\s+', ' ', text).strip()
37     #Converts text to lowercase and splits into words

```

```

38     words = text.lower().split()
39     #Strips punctuation and filters out stopwords
40     words = [word.strip('.,!()?[]:')] for word in words]
41     words = [word for word in words if word not in STOPWORDS]
42     return words #Returns preprocessed words
43
44 def text_to_vector(text):
45     '''Converts text into a vector of word counts so cosine similarity and
rouge score can be calculated.'''
46     words = preprocess(text) #preprocess text
47     return Counter(words) #returns counter of words
48
49 #Inspired by this website: https://datastax.medium.com/how-to-implement-cosine-similarity-in-python-505e8ec1d823
50 def cosine_similarity(vec1, vec2):
51     '''Calculates cosine similarity for each chatbot test question, comparing
with database information.'''
52     intersection = set(vec1.keys()) & set(vec2.keys()) #Finds common words
53     numerator = sum([vec1[x] * vec2[x] for x in intersection]) #Calculates
numerator
54     sum1 = sum([vec1[x]**2 for x in vec1]) #Sum of squares of vector 1
55     sum2 = sum([vec2[x]**2 for x in vec2]) # Sum of squares of vector 2
56     denominator = math.sqrt(sum1) * math.sqrt(sum2) #Calculates denominator
57     return float(numerator) / denominator if denominator else 0.0 #Returns
cosine similarity
58
59 #Inspired by this website: https://thepythoncode.com/article/calculate-rouge-score-in-python
60 def calculate_rouge(text1, text2):
61     '''Calculates Rouge between each response and database information.'''
62     #Uses ROUGE scorer to calculate ROUGE-1, ROUGE-2, and ROUGE-L scores
63     scorer = rouge_scorer.RougeScorer(['rouge1', 'rouge2', 'rougeL'],
use_stemmer=True)
64     scores = scorer.score(text1, text2) #Calculates scores for each test
question and database information
65     return scores #return scores
66
67 def compare_texts(text_pairs):
68     '''Compares each test question and database information for all the
chatbots and calculates similarity and ROUGE scores'''
69     #A list of names for cycling through different names when printing results
for easy identification
70     names_cycle = [
71         "Geo Ed", "Arthurs Seat/The Meadows", "Famous", "Clusters",
"Newbigging", "Arthurs Seat"
72     ]
73     #Specifies that there are six test questions per chatbot.
74     batch_size = 6
75
76     #Goes over each text pair in batches (each test question and relevant
database information)
77     for batch_start in range(0, len(text_pairs), batch_size):

```

```

78     # Select the current batch of text pairs
79     batch_pairs = text_pairs[batch_start:batch_start + batch_size]
80     #sets cosine similarity to 0
81     total_cosine_similarity = 0
82     #sets rouge score to 0
83     total_rouge_scores = {'rouge1': 0, 'rouge2': 0, 'rougeL': 0}
84
85     #For each test question in batch
86     for i, (text1, text2) in enumerate(batch_pairs, start=batch_start):
87         #converts response text to vector
88         vector1 = text_to_vector(text1)
89         #converts database text to vector
90         vector2 = text_to_vector(text2)
91         #calculates cosine similarity
92         cosine_sim = cosine_similarity(vector1, vector2)
93         #calcuates rouge score
94         rouge_scores = calculate_rouge(text1, text2)
95
96         #Determines and prints the chatbot name for each pair
97         if batch_start >= 2 * batch_size:
98             name_suffix = f"Elm {names_cycle[i % batch_size]}"
99         elif batch_start >= batch_size:
100             name_suffix = f"Botsonic {names_cycle[i % batch_size]}"
101         else:
102             name_suffix = names_cycle[i % batch_size]
103
104         print(f"Pair {i+1} ({name_suffix}) Cosine Similarity:
{cosine_sim}")
105         print(f"ROUGE Scores: {rouge_scores}")
106
107         total_cosine_similarity += cosine_sim
108         for key in total_rouge_scores:
109             total_rouge_scores[key] += rouge_scores[key].fmeasure
110
111         #Collects total similarities and scores
112         average_cosine_similarity = total_cosine_similarity / len(batch_pairs)
113         average_rouge_scores = {key: val / len(batch_pairs) for key, val in
total_rouge_scores.items()}
114
115         #Prints results
116         print(f"Average Cosine Similarity for Batch {batch_start // batch_size
+ 1}: {average_cosine_similarity}")
117         print(f"Average ROUGE Scores for Batch {batch_start // batch_size + 1}:
{average_rouge_scores}")
118
119 #The chatbot responses used for analysis
120 text_pairs = [
121
122 #Geo-Bot
123
124 (""""Edinburgh is celebrated for its extensive and varied greenspaces, each of
which has distinct geographical features contributing to the city's landscape

```

places.info/features/featurefirst249.html, Place Name: Arthur's Seat

685 Description: The remains of a volcano that erupted under water during the Carboniferous period, Arthur's Seat is located within Holyrood Park in the centre of Edinburgh. Although only 250m (823 feet) in height, Arthur's Seat is a notable landmark, dominating the city. Known also as the Lion's Head, Arthur's Seat is the highest of a series of peaks which take the form of a crouched lion. Its summit is marked both by a white Ordnance Survey triangulation pillar and a geographical indicator. The indicator gives directions to prominent surrounding landmarks and is a replacement for the original positioned here in 1910, which was cast in bronze and designed by the noted surveyor John Mathieson (1855 – 1945).

686

687 Geologically Arthur's Seat comprises the remains of a basalt lava plug that choked the neck of a volcano which would have been active around 335 million years ago. The action of glaciation has cut into its heart, making it one of the most accessible exposures of an ancient volcano.

688

689 Two stony banks on the east side of the hill represent the remains of an Iron-Age hill-fort and a series of cultivation terraces are obvious above the road just beyond. In 1836, just below the summit, seventeen miniature wooden coffins each containing a carved figure were found in a small cave. Their existence has never been satisfactorily explained; associations with witchcraft have been suggested or perhaps they were memorial to the seventeen victims of the infamous William Burke (1792 – 1829) and William Hare (died c.1860).

690 Coordinates: Latitude: 55.944 , Longitude: 3.1618

691 Link: Arthur's Seat: [https://www.scottish-](https://www.scottish-places.info/features/featurefirst249.html)

692 places.info/features/featurefirst249.html, Place Name: Arthur's Seat

Description: The remains of a volcano that erupted under water during the Carboniferous period, Arthur's Seat is located within Holyrood Park in the centre of Edinburgh. Although only 250m (823 feet) in height, Arthur's Seat is a notable landmark, dominating the city. Known also as the Lion's Head, Arthur's Seat is the highest of a series of peaks which take the form of a crouched lion. Its summit is marked both by a white Ordnance Survey triangulation pillar and a geographical indicator. The indicator gives directions to prominent surrounding landmarks and is a replacement for the original positioned here in 1910, which was cast in bronze and designed by the noted surveyor John Mathieson (1855 – 1945).

693 Coordinates: Latitude: 55.944 , Longitude: 3.1618

694 Link: Arthur's Seat: [https://www.scottish-](https://www.scottish-places.info/features/featurefirst249.html)

695 places.info/features/featurefirst249.html""")

696]

697

698 compare_texts(text_pairs)

699 average_reading_score(text_pairs)

Appendix G: Response Consistency (Jaccard Similarity)

Inspired by this website: <https://www.geeksforgeeks.org/data-science/how-to-calculate-jaccard-similarity-in-python/>

```
def jaccard_similarity(set1, set2):
    '''Calculates Jaccard Similarity between two sets'''
    # Calculates the word similarity of the two sets
    intersection = set1.intersection(set2)
    # Calculates the union (similarity) of the two sets
    union = set1.union(set2)
    # Returns the ratio of the intersection size to the union size (similarity)
    return len(intersection) / len(union)

def text_to_set(text):
    # Converts text to a set of unique words
    return set(text.lower().split())

#Example set of chatbot answers to compare consistency
text_a = '''Edinburgh's greenspaces are diverse and feature various geographical
characteristics:

1. **Holyrood Park**: Home to Arthur's Seat, an ancient volcano, offering hilly
terrain and panoramic views of the city. The park also includes Salisbury Crags,
providing dramatic cliffs and rock formations.

2. **The Meadows**: A flat, expansive parkland with wide open fields and tree-
lined paths, perfect for walking and recreational activities.

3. **Princes Street Gardens**: Nestled between the Old and New Town, these
gardens lie in a valley with gentle slopes, offering lush lawns and floral
displays beneath the backdrop of Edinburgh Castle.

4. **Calton Hill**: Known for its iconic monuments, it provides elevated views
of the city and the Firth of Forth, with grassy areas and rocky outcrops.

5. **Inverleith Park**: Features open, level parkland with views of the city
skyline, a pond, and botanical gardens nearby.

6. **Royal Botanic Garden Edinburgh**: A mix of landscaped gardens and natural
areas, showcasing diverse plant collections on gently sloping terrain.

7. **Blackford Hill**: Offers hilly terrain with rich biodiversity, panoramic
vistas, and the Royal Observatory.

These spaces provide a blend of natural beauty and urban convenience, each with
unique geological and ecological attributes.'''

text_b = '''Edinburgh's greenspaces are diverse and feature various geographical
characteristics:

1. **Holyrood Park**: Home to Arthur's Seat, an ancient volcano, offering hilly
```

terrain and panoramic views of the city. The park also includes Salisbury Crags, providing dramatic cliffs and rock formations.

2. **The Meadows**: A flat, expansive parkland with wide open fields and tree-lined paths, perfect for walking and recreational activities.

3. **Princes Street Gardens**: Nestled between the Old and New Town, these gardens lie in a valley with gentle slopes, offering lush lawns and floral displays beneath the backdrop of Edinburgh Castle.

4. **Calton Hill**: Known for its iconic monuments, it provides elevated views of the city and the Firth of Forth, with grassy areas and rocky outcrops.

5. **Inverleith Park**: Features open, level parkland with views of the city skyline, a pond, and botanical gardens nearby.

6. **Royal Botanic Garden Edinburgh**: A mix of landscaped gardens and natural areas, showcasing diverse plant collections on gently sloping terrain.

7. **Blackford Hill**: Offers hilly terrain with rich biodiversity, panoramic vistas, and the Royal Observatory.

These spaces provide a blend of natural beauty and urban convenience, each with unique geological and ecological attributes.'

text_c = '''Edinburgh's greenspaces boast diverse geographical features:

1. **Holyrood Park**: Features Arthur's Seat, an ancient volcano, and Salisbury Crags, offering rugged terrain and panoramic city views.

2. **The Meadows**: A flat, expansive park with open fields and tree-lined paths ideal for leisure activities.

3. **Princes Street Gardens**: Situated in a valley beneath Edinburgh Castle, with gentle slopes and lush lawns.

4. **Calton Hill**: Known for its iconic monuments and elevated city views, featuring grassy areas and rocky outcrops.

5. **Inverleith Park**: Offers level parkland with views of the city skyline and a central pond.

6. **Royal Botanic Garden Edinburgh**: Combines landscaped gardens with natural areas on gently sloping terrain.

7. **Blackford Hill**: Provides hilly terrain with panoramic vistas and rich biodiversity.

These greenspaces blend natural beauty and urban convenience, each with unique geological features.'

#Converts texts to sets of words

set_a = text_to_set(text_a)

```
75 set_b = text_to_set(text_b)
76 set_c = text_to_set(text_c)
77
78 # Calculates Jaccard similarities between the three answers
79 jaccard_ab = jaccard_similarity(set_a, set_b)
80 jaccard_bc = jaccard_similarity(set_b, set_c)
81 jaccard_ac = jaccard_similarity(set_a, set_c)
82
83 # Calculates total similarity score
84 total_similarity_score = (jaccard_ab + jaccard_bc + jaccard_ac) / 3
85
86 # Print results
87 print(f"Jaccard Similarity between Text A and B: {jaccard_ab}")
88 print(f"Jaccard Similarity between Text B and C: {jaccard_bc}")
89 print(f"Jaccard Similarity between Text A and C: {jaccard_ac}")
90 print(f"Total Similarity Score: {total_similarity_score}")
```

Appendix H: User Feedback Results

Responses Overview Active

Responses

9

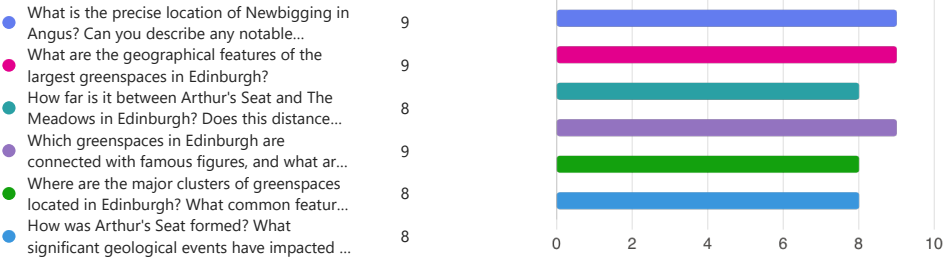
Average Time

31:53

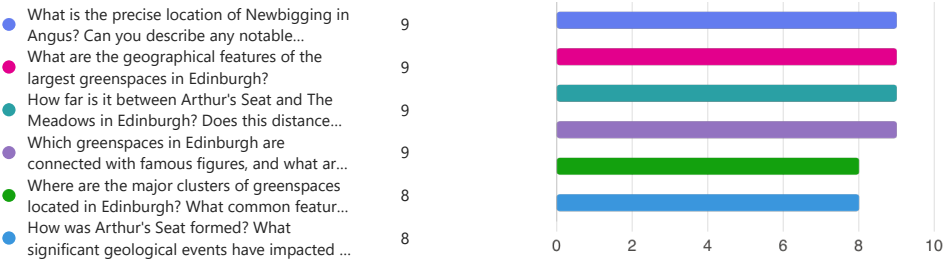
Duration

67 Days

1. Did the chatbot produce a detailed and geographically comprehensive response to each query? Select if yes.



2. Was the information retrieved by the chatbot relevant to each query? Select if yes.



3. Did the chatbot ever fail to successfully answer a query by providing an incorrect response or no response at all?



4. For what question(s) did it fail and how?

3

Responses

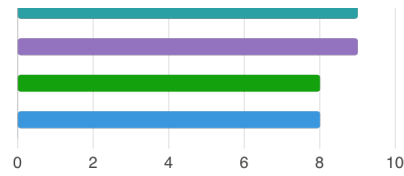
Latest Responses

...

5. Was the generated map accurate for each query?

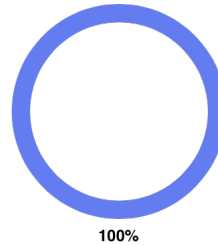


- How far is it between Arthur's Seat and The Meadows in Edinburgh? Does this distance... 9
- Which greenspaces in Edinburgh are connected with famous figures, and what ar... 9
- Where are the major clusters of greenspaces located in Edinburgh? What common featur... 8
- How was Arthur's Seat formed? What significant geological events have impacted ... 8



6. How clear and accessible were the maps to navigate?

- Extremely 9
- Somewhat 0
- Adequate 0
- Difficult 0
- Impossible 0



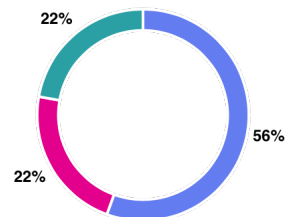
7. What could be improved about the mapping?

3
Responses

Latest Responses
...

8. Were the Gazetteer for Scotland links accurate?

- Always 5
- Very often 2
- Sometimes 2
- Rarely 0
- Never 0



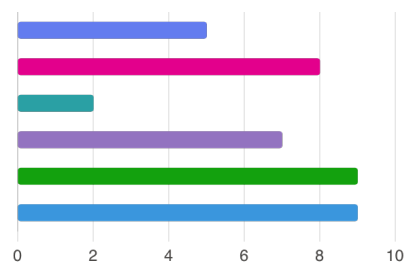
9. What could be improved about this chatbot?

4
Responses

Latest Responses
...

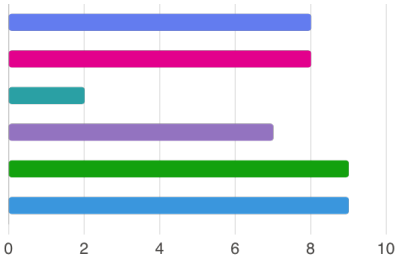
10. Did the chatbot produce a detailed and geographically comprehensive response to each query? Select if yes.

- What is the precise location of Newbigging in Angus? Can you describe any notable... 5
- What are the geographical features of the largest greenspaces in Edinburgh? 8
- How far is it between Arthur's Seat and The Meadows in Edinburgh? Does this distance... 2
- Which greenspaces in Edinburgh are connected with famous figures, and what ar... 7
- Where are the major clusters of greenspaces located in Edinburgh? What common featur... 9
- How was Arthur's Seat formed? What significant geological events have impacted ... 9



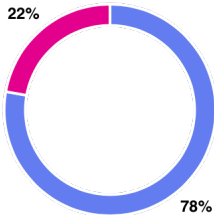
11. Was the information retrieved by the chatbot relevant to each query? Select if yes.

- What is the precise location of Newbigging in Angus? Can you describe any notable... 8
- What are the geographical features of the largest greenspaces in Edinburgh? 8
- How far is it between Arthur's Seat and The Meadows in Edinburgh? Does this distance... 2
- Which greenspaces in Edinburgh are connected with famous figures, and what ar... 7
- Where are the major clusters of greenspaces located in Edinburgh? What common featur... 9
- How was Arthur's Seat formed? What significant geological events have impacted ... 9



12. Did the chatbot ever fail to successfully answer a query by providing an incorrect response or no response at all?

- Yes 7
- No 2



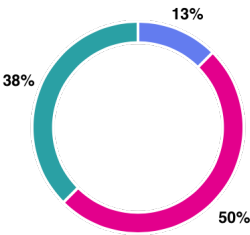
13. For what question(s) did it fail and how?

7
Responses

Latest Responses
"Question 3"
"How far is it between Arthur's Seat and The Meadows in Edinburgh? Does th..."
...

14. Were the Gazetteer for Scotland source links accurate?

- Always 1
- Very often 4
- Sometimes 3
- Rarely 0
- Never 0



15. What could be improved about this chatbot?

4
Responses

Latest Responses
"The visuals and detail included in this chatbot was not as thorough. This cou..."
...

16. Which chatbot performed the strongest overall?

- 1 Geo-Chatbot
- 2 Alternative Chatbot



3

They both performed equally well

17. Which chatbot gave the most accurate responses?

1 Geo-Chatbot

2 Alternative Chatbot

3 They both performed equally well

18. Which chatbot was the easiest to use?

1 Alternative Chatbot

2 Geo-Chatbot

3 They both performed equally well

19. Which chatbot did you prefer overall?

1 Geo-Chatbot

2 Alternative Chatbot

3 They both performed equally well

20. Any final comments?

6

Responses

Latest Responses

"I liked the visual mapping of the GeoBot, and thought this worked well. The ... "

...

Appendix I: Responses To the Tested ChatBot Questions

Question One: What are the geographical features of greenspaces in Edinburgh?

General understanding of Scottish Places (Place, Environment)

GeoBot

Edinburgh, Scotland's capital, is renowned for its abundant and varied greenspaces, which include parks, hills, and gardens that significantly contribute to the city's landscape. Important geographical features of these greenspaces include:

1. **Volcanic Hills:** Several of Edinburgh's hills, such as Arthur's Seat, Calton Hill, and Castle Rock, are ancient volcanic formations. Arthur's Seat, the most prominent, provides a dramatic backdrop to the city and is a popular destination for walkers and nature enthusiasts.
2. **Water Features:** The city is enriched with water features, like the Water of Leith, which runs through several areas providing natural beauty and habitat for wildlife. Duddingston Loch is another significant body of water located near Holyrood Park, offering a tranquil setting away from urban activity.
3. **Wooded Areas:** Areas like Corstorphine Hill and the Royal Botanic Garden offer wooded escapes with a variety of tree species and plants, ideal for biodiversity and a retreat into nature.
4. **Formal Gardens and Designed Landscapes:** The Royal Botanic Garden Edinburgh is a prime example, featuring beautifully designed landscapes, a wide range of plant species, and various thematic gardens.
5. **Recreational Greenspaces:** The Meadows and Inverleith Park are central recreational greenspaces where sports, walking, and community events are routinely enjoyed by locals and visitors alike.
6. **Coastal Features:** The beaches at Cramond and Silverknowes to the north of the city provide sandy expanses and views over the Firth of Forth, blending marine and terrestrial environments.
7. **Golf Courses:** With around twenty golf courses, Edinburgh is well provided for in terms of landscaped spaces that offer not only recreational opportunities but also serve as significant green lungs for the city.

These diverse geographical features play a crucial role in defining the character and environmental quality of Edinburgh, making the city both aesthetically pleasing and ecologically rich. Coordinates for Edinburgh: Latitude: 55.9523, Longitude: -3.1882. Follow this link for more information about Edinburgh.

No-Code Chatbot Botsonic

Edinburgh is home to several greenspaces, each with unique geographical features. One of the most notable is Holyrood Park, which spans 260 hectares and includes Arthur's Seat, an ancient volcano offering stunning views of the city. The park also features Salisbury Crags, dramatic cliffs formed from volcanic rock, and several lochs, including Duddingston Loch, a bird sanctuary.

Another significant greenspace is The Meadows, a large public park known for its open fields and tree-lined paths. It is situated near the city center and is a popular spot for picnics and outdoor activities. Additionally, the Royal Botanic Garden showcases a variety of plant species and includes themed areas like the alpine rock garden, enhancing its geographical diversity.

These greenspaces not only provide recreational opportunities but also contribute to the ecological richness of Edinburgh, making them essential for both residents and visitors.

Base GPT-4

Edinburgh's greenspaces are diverse and include a variety of geographical features:

1. **Arthur's Seat and Holyrood Park**: A dormant volcano and surrounding landscapes with rugged hills, cliffs, and lochs, offering panoramic views of the city.
2. **The Meadows**: A large, flat parkland adjacent to the city center, popular for recreational activities and surrounded by tree-lined paths.
3. **Calton Hill**: Known for its historic monuments and excellent views of the city and the Firth of Forth. The hill features grassy slopes and rocky areas.
4. **Royal Botanic Garden**: Situated on slightly sloping terrain, it includes diverse plant collections, ornamental landscapes, and a rock garden.
5. **Inverleith Park**: Located near the Botanic Garden, it features expansive playing fields, a pond, and gentle hills.
6. **Water of Leith Walkway**: This riverside path runs through the city, flanked by woodland and steep embankments, providing a natural corridor through urban areas.
7. **Blackford Hill and Hermitage of Braid**: Comprising hilly terrain with woods, open grassland, and a nature reserve, offering views of the Pentland Hills.
8. **Pentland Hills Regional Park**: Situated to the south of the city, characterized by rolling hills and moorland, popular for hiking and outdoor pursuits.

Question Two: How far is it between Arthur's Seat and the Meadows in Edinburgh? Does this distance vary depending on the mode of transport used?

Concept of Nearness and Inferencing from Coordinates (Space)

GeoBot

The distance between Arthur's Seat and the Meadows in Edinburgh is approximately 1.6 miles. This calculation assumes a straight-line distance.

The actual distance can vary depending on the mode of transport used due to the different routes that may be taken. Walking paths might follow a more direct route compared to roads that vehicles must use, which might need to navigate around buildings, through intersections, and follow traffic regulations.

- Coordinates for Arthur's Seat: Latitude: 55.944, Longitude: -3.1618. Follow this link for more information about Arthur's Seat.

- Coordinates for the Meadows: Latitude: 55.9395, Longitude: -3.1827. Follow this link for more information about the Meadows.

No-Code Chatbot Botsonic

I apologize, but I don't have enough information in my knowledge base to answer that question. Please feel free to ask something else I might be able to help with.

Base GPT-4

The distance between Arthur's Seat and The Meadows in Edinburgh is approximately 1.5 to 2 miles, depending on the route taken. This distance can vary slightly based on the path or mode of transport:

- ****Walking****: Walking paths and pedestrian routes may include gradual inclines and scenic detours, typically taking around 30 to 40 minutes.

- ****Cycling****: Cycling routes might be slightly longer due to designated bike paths and can take about 10 to 15 minutes.

- ****Driving****: By car or public transport, it can take about 10 to 15 minutes, depending on traffic conditions.

Each mode offers a unique perspective of the city and its landscapes.

Question Three: Which greenspaces in Edinburgh are connected with famous figures, and what are their historical links?

Concept of Associations (Interconnection)

GeoBot

Edinburgh is adorned with several greenspaces connected to renowned figures, each harbouring rich historical ties:

1. Dunbar's Close Garden:

- Description: This serene garden, planted with flowers and herbs, is a haven tucked away near the bustling Royal Mile. It holds associations with the visionary town planner, Sir Patrick Geddes.

- Coordinates for Dunbar's Close Garden: Latitude: 55.9521, Longitude: -3.1789. Follow this link for more information about Dunbar's Close Garden.

2. Grannies Green:

- Description: Grannies Green, a small garden set on steep ground between Kings Stables Road and Johnston Terrace, was transformed into a commemorative site for Sir Patrick Geddes. The green features wildflowers, a herb garden, and fruit trees. Interpretation boards and ceramic panels celebrate Geddes' contributions to urban planning.

- Coordinates for Grannies Green: Latitude: 55.9475, Longitude: -3.1982. Follow this link for more information about Grannies Green.

3. Royal Botanic Garden Edinburgh:

- Description: Initially established near Holyrood Abbey as a 'physic garden' by Sir Robert Sibbald and Andrew Balfour in 1671, the garden was later enhanced and relocated by Dr. John Hope. It thrived under various caretakers and now stands as an internationally recognized research and education centre.

- Coordinates for Royal Botanic Garden Edinburgh: Latitude: 55.9658, Longitude: -3.2073. Follow this link for more information about Royal Botanic Garden Edinburgh.

These greenspaces not only provide tranquility and beauty in Edinburgh but also serve as educational and commemorative sites linked to figures who have shaped both the city's and the nation's history.

No-Code ChatBot Botsonic

In Edinburgh, several greenspaces are connected with notable historical figures, showcasing their contributions and legacies.

- Grannies Green: This small garden commemorates Sir Patrick Geddes, a renowned town planner and environmentalist. Originally a space for drying laundry, it was transformed in 2012 to honor Geddes, featuring wildflowers, a herb garden, and interpretation boards celebrating his life.

- **Dunbar's Close Garden:** Located near the Canongate, this garden reflects the design of a 17th-century Burghal garden. It was commissioned by the Mushroom Trust and is associated with Geddes, providing a peaceful retreat filled with flowers and herbs.
- **St Andrew Square Garden:** This garden, surrounding the Melville Monument, has historical ties to figures like Charles Dickens, who visited in 1834, and Henry Dundas, the 1st Viscount Melville, in whose memory the monument was erected. The garden has evolved into a public space, enhancing its historical significance.

These greenspaces not only offer beauty but also tell the stories of influential individuals in Edinburgh's history.

Base GPT-4

Edinburgh's greenspaces are rich in historical connections with famous figures:

1. **Holyrood Park and Arthur's Seat**:

- Linked to King David I, who founded Holyrood Abbey in 1128. The parkland has been part of the royal estate for centuries.

2. **Princes Street Gardens**:

- Situated in the heart of the city, these gardens are associated with Sir Walter Scott, whose monument stands prominently there. The gardens were created from the drained Nor Loch, which Scott often referenced.

3. **Calton Hill**:

- Notable for its monuments dedicated to figures like Admiral Lord Nelson and philosopher Dugald Stewart. The National Monument is sometimes called "Edinburgh's Disgrace," intended to honor the dead of the Napoleonic Wars, including prominent Scots.

4. **The Meadows**:

- Historically used for public events and sports; notable figures such as J.K. Rowling have made appearances at events here.

5. **Royal Botanic Garden**:

- Formerly directed by John Hope, an important figure in botany. The garden has attracted numerous botanists and scholars over time.

6. **Blackford Hill and Hermitage of Braid**:

- Associated with scientific exploration, as figures like James Hutton, the "father of geology," studied the area's rock formations.

These greenspaces not only provide natural beauty but also embody Edinburgh's rich historical tapestry.

Question Four: Where are the major clusters of greenspaces located in Edinburgh? What common features or characteristics do these greenspaces have?

Concept of Clustering (Interconnection, Space)

GeoBot

In Edinburgh, major clusters of greenspaces are prominently located in various parts of the city, each offering unique features and extensive recreational opportunities:

1. Central Edinburgh: This area includes greenspaces such as Princes Street Gardens, The Meadows, and Holyrood Park. These are centrally located, making them easily accessible for both locals and tourists. Holyrood Park, with Arthur's Seat as its crowning feature, offers extensive hillwalking opportunities right in the city centre.

- Coordinates for Princes Street Gardens: Latitude: 55.9504, Longitude: -3.1993. Follow this link for more information about Princes Street Gardens.

2. North Edinburgh: This cluster includes Inverleith Park and the Royal Botanic Garden Edinburgh. These spaces are known for their well-maintained gardens, sports facilities, and scenic views. The Royal Botanic Garden Edinburgh is not only a recreational area but also an internationally important scientific centre for botanical research.

- Coordinates for Royal Botanic Garden Edinburgh: Latitude: 55.9658, Longitude: -3.2073. Follow this link for more information about Royal Botanic Garden Edinburgh.

3. East Edinburgh: Leith Links and Portobello Beach offer open greenspace and recreational areas in the eastern part of the city. Leith Links has historical significance related to the development of golf in Scotland.

- Coordinates for Leith Links: Latitude: 55.9714, Longitude: -3.1650. Follow this link for more information about Leith Links.

Common Features of Edinburgh's Greenspaces:

- Accessibility: Most of these spaces are centrally located or well-connected through public transport, making them accessible to a wide audience.

- Cultural and Historical Significance: Many of these greenspaces are tied to the city's rich history, either through historical events or associations with notable figures.

- Biodiversity and Natural Beauty: These areas are maintained not only for recreational purposes but also for conservation, supporting diverse plant and animal life.

- Recreational Facilities: From sports pitches and playgrounds to walking, cycling paths, and nature reserves, these greenspaces offer a variety of activities.

These clusters and characteristics highlight Edinburgh's commitment to maintaining green areas that enhance urban life and preserve natural and cultural heritage.

No-Code ChatBot Botsonic

Edinburgh boasts several major clusters of greenspaces, each with unique characteristics. One of the most prominent is Holyrood Park, which covers 260 hectares and features Arthur's Seat, an ancient volcano offering panoramic views of the city. This park is known for its rugged terrain and natural beauty, making it a popular spot for hiking and outdoor activities.

Another significant area is The Meadows, located near the city center. This park provides 14.5 hectares of open space, ideal for picnics, sports, and community events. It features tree-lined paths and is historically linked to Sir Thomas Hope, who transformed the area in the 18th century.

Other notable greenspaces include Craigmillar Castle Jubilee Park and the Royal Botanic Garden, which showcases a diverse range of plant species. These greenspaces not only enhance the city's landscape but also serve as vital recreational areas for residents and visitors alike.

Base GPT-4

In Edinburgh, major clusters of greenspaces can be found in several areas:

1. **City Centre:**

- **Princes Street Gardens:** Located in the heart of the city, featuring ornamental gardens and historic monuments.
- **Calton Hill:** Known for its panoramic views and iconic monuments.

2. **South of the City:**

- **The Meadows:** A large, flat park popular for recreation and community events.
- **Blackford Hill and Hermitage of Braid:** Offers woodland trails and views, ideal for hiking and nature enthusiasts.

3. **East of the City:**

- **Holyrood Park and Arthur's Seat:** A vast area with a dormant volcano, rugged terrain, and lochs, perfect for hiking and taking in the city's skyline.

4. **North of the City:**

- **Royal Botanic Garden:** Features diverse plant collections, a rock garden, and landscaped areas.
- **Inverleith Park:** Adjacent to the Botanic Garden, offering playing fields, a pond, and open spaces.

Common Features:

- **Biodiversity:** Many of these greenspaces are rich in flora and fauna.
- **Historic Significance:** Sites like Calton Hill and Princes Street Gardens are steeped in history.
- **Recreational Facilities:** Most offer paths for walking, running, and cycling, as well as areas for sports.

- ****Scenic Views****: Particularly from elevated spots like Arthur's Seat and Calton Hill.

These greenspaces provide a blend of natural beauty, recreational opportunities, and historical context.

Question Five: What is the precise location of Newbigging in Angus? Can you describe any landmarks or features nearby?

Concept of Overlapping and Ambiguity (Space, Interpretation)

GeoBot

Newbigging in Angus includes multiple locations with the same name, each distinguished by different surroundings and landmarks:

1. Newbigging near Newtyle

Description: This small hamlet is located close to the western boundary of Angus, near Newtyle. It is notable for its proximity to other rural communities such as Ardler and Meikle in Perth and Kinross.

Coordinates for Newbigging, Angus: Latitude: 56.5645, Longitude: -3.1617. Follow this [link](#) for more information about Newbigging, Angus.

2. Newbigging near Monikie

Description: This village in southeast Angus expanded in the 1950s with the construction of public housing. Nearby, stone is quarried, and historical sites like the Carlungie and Ardestie souterrains are located to the east and south, respectively.

Coordinates for Newbigging, Angus: Latitude: 56.5142, Longitude: -2.8189. Follow this [link](#) for more information about Newbigging, Angus.

3. Newbigging near Tealing

Description: Positioned in south Angus, this hamlet is immediately east of the A90 trunk road, close to Tealing and north of Dundee. Its location near main transport routes highlights its connectivity to larger urban areas.

Coordinates for Newbigging, Angus: Latitude: 56.5268, Longitude: -2.9395. Follow this [link](#) for more information about Newbigging, Angus.

Each location provides unique insights into the rural life of Angus with its distinctive landmarks and cultural heritage. Please specify if you need information about one particular Newbigging or further details on the nearby features.

No-Code Chatbot Botsonic

Newbigging is a small village located in southeast Angus. It lies approximately 1¼ miles (2 km) south of Monikie and about 2 miles (3 km) east-southeast of Wellbank. Additionally, it is situated immediately to the east of the A90 trunk road, about half a mile (1 km) southeast of Tealing and 4 miles (6.5 km) north of Dundee. In terms of notable features, Newbigging has a history tied to public housing development in the 1950s. Nearby, stone is quarried, and there are ancient souterrains located to the east at Carlungie and to the south at Ardestie. These features contribute to the village's unique character and historical significance.

Base GPT-4

Newbigging is a small village located in Angus, Scotland, near Monifieth. It lies approximately 5 miles northeast of Dundee.

Notable landmarks and features in or around Newbigging include:

Monikie Country Park: Situated close by, it offers outdoor activities like walking and water sports.

Barry Mill: A historical water-powered mill located a few miles east of Newbigging.

Panmure Testimonial: A prominent stone monument visible from the village, located in the nearby parklands. The area is primarily rural, with scenic landscapes and farmland surrounding the village.

Question Six: How was Arthur's Seat formed? What significant geological events have impacted it over time?

Concept of Time

GeoBot

Arthur's Seat is a prominent geological feature located in the heart of Edinburgh, formed through a series of ancient volcanic activities. The main component of Arthur's Seat is a basalt lava plug from an extinct volcano. This volcano was active approximately 335 million years ago during the Carboniferous period. Over time, the surrounding softer rock was eroded away, leaving the hard basalt exposed.

Significant geological events that have shaped Arthur's Seat include the movement of tectonic plates and glaciation. The area was once subjected to intense volcanic activity due to the movement of the Earth's crust. Following the cessation of volcanic activity, the region underwent extensive glaciation during the last ice age. This glaciation sculpted the landscape, carving out the crags and tails characteristic of Arthur's Seat today. These geological processes have left Arthur's Seat as one of the most accessible exposures of ancient volcanic structure in the region.

Coordinates for Arthur's Seat: Latitude: 55.944, Longitude: -3.1618. Follow this link for more information about Arthur's Seat.

No-Code ChatBot Botsonic

Arthur's Seat is formed from the remains of a volcano that erupted underwater during the Carboniferous period, approximately 335 million years ago. This ancient volcano is characterized by a basalt lava plug that choked its neck, creating the prominent hill we see today. Over time, glaciation has significantly shaped its landscape, cutting into the volcanic rock and exposing its geological features, making it one of the most accessible examples of an ancient volcano.

The geological history of Arthur's Seat is marked by various events, including the action of glaciation, which has sculpted its current form. The area also contains remnants of an Iron-Age hill-fort and cultivation terraces, indicating human activity over the centuries. These features, combined with the volcanic origins, contribute to the unique geological significance of Arthur's Seat within Holyrood Park, a vital resource for studying both geology and archaeology. Let me know if you have any more questions!

Base GPT-4

Arthur's Seat was formed through a series of significant geological events:

1. ****Volcanic Activity****: Around 350 million years ago, during the Carboniferous period, volcanic eruptions created a series of volcanic hills, including Arthur's Seat. The main peak is the remains of a volcanic plug.

2. **Glaciation**: During the Ice Ages, glaciers scoured the landscape, shaping the hill and surrounding area. This glaciation smoothed the volcanic rock and created features like the nearby Salisbury Crags.

3. **Erosion**: Over time, wind and rain erosion further shaped the landscape, exposing various rock formations and contributing to the hill's rugged appearance.

These geological processes have left Arthur's Seat as a prominent natural feature in Edinburgh, offering insights into the city's ancient geological history.